National Defence Défense nationale

DRES SUFFIELD REPORT

NO. 589

THE EFFECT OF SENSORY INPUT ON TRAJECTORIES GENERATED BY RECURRENT NEURAL NETWORKS

by

Simon A. Barton

PCN 031SC

January 1993

DEFENCE RESEARCH ESTABLISHMENT SUFFIELD : RALSTON : ALBERTA

DEFENSIO PER SCIENTIAM

DEFENCE RESEARCH ESTABLISHMENT SUFFIELD
RALSTON, ALBERTA

SUFFIELD REPORT NO. 589

THE EFFECT OF SENSORY INPUT ON TRAJECTORIES GENERATED BY RECURRENT NEURAL NETWORKS

by

Simon A. Barton

PCN 031SC

ABSTRACT

We show that the iterative signals generated by a simple recurrently connected network of only 3 sigmoidal nodes become stable, periodic or chaotic, depending on the magnitude of a global parameter (s) that controls the steepness of the node responses. High values of s produce chaotic signals, and low values lead to stable signals, i.e. unchanging from cycle to cycle. The transition from stability, through periodicity to chaos is shown to be consistent with Feigenbaum's model of critical nonlinear systems.

For large networks that are recurrently connected, the stability also depends critically on the node connection weights and biases. When these are fixed randomly, it is possible to find large systems that also show a transition from stability to chaos by the adjustment of s. When the weights and biases are allowed to evolve by a modified Hebbian rule, which depends on the internal state of the network, we find that the system becomes stable or chaotic, again depending on s, but does not appear to exhibit periodicity.

We have developed a software simulation of a machine whose motion in two dimensions is controlled entirely by the iterative activity of a recurrently connected neural network. The activities of some randomly chosen nodes drive motor responses that control the movement. In the absence of external input, the machine can be made to spiral into inactivity or to follow a chaotic trajectory. If the simulated environment contains one or more emitting sources, and the machine carries sensors whose responses are fed into the network, the motion is influenced by the sensory input. Under a Hebbian-type learning scheme, we show that the machine is strongly attracted to an emitting source when s is not in the chaotic regime. The motion of the machine in the region of a source is characterized by cycles of erratic attraction and repulsion, and is highly reminiscent of the motion of a moth near a light.

It is possible that recurrent networks may be used to develop navigation and obstacle avoidance capabilities that are self-learned in an autonomous land vehicle.

RÉSUMÉ

Nous montrons que les signaux itératifs produits par un réseau simple à connexions récurrentes de seulement trois noeuds sigmoïdaux deviennent stables, périodiques ou chaotiques, selon la grandeur d'un paramètre global (s) de stabilité des réponses des noeuds. Selon que la valeur de s est élevée ou faible, la réponse est chaotique ou stable, c'est-à-dire constante d'un cycle à l'autre. La transition d'un état stable à un état périodique, puis chaotique, est en accord avec le modèle des systèmes non linéaires critiques de Feigenbaum.

Dans les grands réseaux à connexions récurrentes, la stabilité dépend aussi grandement de la pondération et de la polarisation des connexions entre les noeuds. Lorsque ces paramètres sont fixés au hasard, il est possible de trouver de grands systèmes qui comportent aussi une transition de l'état stable à l'état chaotique selon la valeur attribuée à s. Lorsque nous laissons la pondération et la polarisation évoluer suivant une règle Hebbienne modifiée, qui dépend de l'état interne du réseau, nous trouvons que le système devient stable ou chaotique, toujours selon la valeur de s, sans présenter aucune périodicité.

Nous avons mis au point une simulation par logiciel d'une machine dont le mouvement en deux dimensions est entièrement commandé par l'activité itérative d'un réseau neuronal à connexions récurrentes. Les activités de certains noeuds choisis au hasard déterminent les réponses qui commandent le mouvement. En l'absence d'intrants externes, la machine peut être amenée à spiraler vers un état d'inactivité ou à suivre une trajectoire chaotique. Si le milieu simulé renferme une ou plusieurs sources émettrices et que la machine porte des capteurs dont les réponses sont introduites dans le réseau, le mouvement est fonction des intrants sensoriels. A l'aide d'un modèle d'apprentissage de type Hebbien, nous montrons que la machine est fortement attirée vers une source émettrice lorsque s n'est pas en régime chaotique. Le mouvement de la machine au voisinage d'une source est caractérisé par des cycles d'attraction et de répulsion erratiques, et rappelle beaucoup celui d'un papillon de nuit près d'une lampe.

Il est possible que des réseaux récurrents puissent servir à mettre au point des moyens de navigation et d'évitement des obstacles qui sont auto-appris dans un véhicule terrestre autonome.

Table of Contents

List of Figures

List of Tables

1. Introduction

Part of the work performed by The Advanced Guidance Concepts Group at DRES is research on radical methods for developing machine intelligence. The long-term goal is to develop navigation, obstacle avoidance and object recognition capabilities in an adaptive, intelligent, autonomous land vehicle.

Over the last ten years, artificial neural network (ANN) techniques have shown promise in the areas of nonlinear mapping, prediction, pattern recognition and adaptive nonlinear control. Based on the performance of biological systems, iterative, recurrently connected networks should be capable of controlling the motion of a dynamic system in a changing environment.

The work presented here is the beginning of a bottom-up approach to machine intelligence based on the self-organizing properties of recurrently connected ANNs We wish to investigate the development and characteristics of the behaviour of a machine that is controlled by a recurrent network when it is allowed to evolve in an environment that provides input to the network through sensors. Output from the network will drive motor responses of the machine. No human criteria will influence the behaviour of the machine; i.e. any learning that occurs will be unsupervised.

Our ANNs consist of many identical nonlinear processing nodes that are randomly interconnected. Each node sums the weighted input from many other nodes, adds a constant bias (or offset), performs a nonlinear transformation (sigmoidal) and passes the output to many other nodes. These systems are connected *recurrently*; i.e. there is continuous feedback. In the absence of changing external input, these are closed, iterative, nonlinear systems.

Iterative, nonlinear systems are known to be capable of generating periodic and chaotic signals.

Consider a nonlinear function of a variable x, whose shape is determined by a parameter λ; i.e. $f(x;\lambda)$. Feigenbaum [1] showed that for a large class of such functions, values generated iteratively exhibit a periodicity that doubles as the magnitude of λ increases past well-defined values. Beyond a critical value of λ, iterative values of f become aperiodic and are said to be chaotic.

Iterative signal generation may be written

$$x_{i+1} = f(x_i; \lambda) \tag{1.1}$$

Thus if $x_{i+1} = x_i$ the period is 1, and x_i is a fixed point of f. If $x_{i+2} = x_i$ the period is 2, if $x_{i+4} = x_i$ the period is 4, and so on. If λ_n is the value of λ for which the n-th periodicity of x first occurs, Feigenbaum showed that λ_n converges geometrically in n to λ_∞, for which the system is aperiodic. He defined

$$\delta_n = \frac{\lambda_{n+1} - \lambda_n}{\lambda_{n+2} - \lambda_{n+1}} \tag{1.2}$$

and found that δ_n approaches a universal constant

$$\lim_{n\to\infty} \delta_n = \delta = 4.6692.. \tag{1.3}$$

for *all* nonlinear systems that exhibit period doubling.

A recurrently connected network of nonlinear processing nodes can therefore be expected to generate periodic and chaotic signals under certain circumstances.

Recurrent artificial neural networks are often referred to as Hopfield nets because of the pioneering work done by J.J. Hopfield. His original model [2] employed a completely connected set of simple linear threshold elements (LTEs) as nodes. The output of each node was either on or off (0 or 1), depending on whether or not the sum of the weighted input signals was above the node threshold. The signals passing between nodes were modified by a matrix of weights, which were iteratively adjusted to minimize a Lyapunov function. Hopfield's later work [3] employed sigmoidal nodes. The output of a sigmoidal node is a continuous, bounded, nonlinear, monotonically-increasing function of the summed input. For both types of network, Hopfield showed that stable network states, giving a constant output, were possible.

Bruck [4] proved that Hopfield nets with LTEs *always* converge to stable or cyclic states of period 2 or 4, if the matrix of connection weights between the nodes is symmetric or antisymmetric. He termed these "the only interesting cases", but also showed that for an arbitrary weight matrix there exists a network of order n that has a cycle of period $2^{n/3}$. For example, with 30 LTE nodes there exists a network of period 1024. Bruck did not consider sigmoidal nodes.

The presence and role of periodicity and chaos in biological systems has been discussed by Freeman and coworkers [5]-[7], and a simulation of biological neurons by Chay [8] also showed the presence of complex oscillations and chaos. A mathematical proof of the existence of period-doubling to chaos in sigmoidal neurons was also given by Wang [9]. The generation of chaotic behaviour in neural networks has been widely discussed [10]-[16].

In the first part of this work (Chapter 2) we show .hat the generation of stable, periodic and chaotic signals in recurrent ANNs can be controlled by a single global parameter s, which determines the steepness of the node response functions. A simple recurrent network of only 3 sigmoidal nodes can exhibit cyclical behaviour, with period doubling observed from 1,2,.. 1024 as s increases. For higher values of s, chaotic behaviour results. From the values of s at which each period doubling occurs, we obtain an estimate (4.669) of Feigenbaum's universal constant δ, showing that the process does indeed lead to a chaotic system. A similar study [16] of a system of 3 sigmoidal nodes with delay in the signal transmission, showed that periodicity and chaos can also result from connection weight and bias modifications.

Beyond the first chaotic region in our 3-node system, there exist smaller regions of s, for which cyclic behaviour and period doubling reoccur. We have observed regions of periodicity 7, 14, 28, 56,.., and also 9, 18, 36, 72.., and 8, 16, 32, 64.. For even higher values of s, cycles of unpredictable period occur (3, 6, 5, 13, 4..). This erratic behaviour is shown to be due to near discontinuities that arise when the sigmoids become almost step functions (quasi-LTEs).

We also show that the stability of large networks with fixed weights and biases can be controlled by the adjustment of the steepness parameter, s. For dynamic networks (Chapter 3), in which the weights and biases are allowed to evolve by rules that depend on the internal state of the network, the stablity may again be controlled. When the activity of several nodes is used to define a trajectory in two-dimensional space, we find that in the absence of external input, the trajectories either spiral in to a fixed point or define bounded chaotic regions.

Thus we understand quite well the behaviour of a simulated recurrent neural network, evolving by internal rules, that may be driving the movement of a machine when it receives no sensory input. In the second part of Chapter 3 we then examine the effect of input to the network from sensors that constantly receive intensity from a simulated emitting source, such as a light. The intensity, and hence the input to the network, varies as the position of the machine, in a true dynamic simulation. The trajectory that such a machine will follow is shown to be dramatically affected by the presence of sensory input.

Finally, in Chapter 4 we discuss the implication of these results for the possible development of machine intelligence, and capabilities of obstacle avoidance and navigation in an autonomous land vehicle.

2. Recurrent Networks

We consider networks that are composed of highly connected processing nodes, the output of each node n being a sigmoidal function f_n of the total input signal x_n. We define

$$f_n(x_n; s, x_{0n}) \equiv [1 + \exp(-s(x_n - x_{0n}))]^{-1} \tag{2.1}$$

The function f_n is parametric in a steepness factor s and an offset x_{0n}. Figure 2.1 shows the sigmoidal curves for $s = 4.0$ and 8.0, with an offset of 0.5. The offset plays the same role as a bias or threshold. We will also refer to the node output as its *activity*.

The total input signal to the n-th node at cycle t is

$$x_n^{(t)} = \sum_{m=1}^{N} W_{nm} f_m^{(t-1)} \tag{2.2}$$

i.e. it is the weighted sum of the outputs from the previous cycle of all the N nodes in the network. If the network is not completely connected, some of the weights will be zero. Also, the diagonal elements of the weight matrix $\mathbf{W}$ will be zero for nodes that do not have direct feedback.

In this work, we set $W_{nn} = 0$ for all n, and we may normalise the sum of the absolute values of the weights at each node in order to limit the total input; i.e.

$$\sum_m |W_{nm}| \equiv c \tag{2.3}$$

where c is a constant.

In principle, the steepness parameter s could be different for each node, but in this work the same value of s is applied to every node.

The iterative output of the network in matrix form is

$$\mathbf{f}^{(t)} = [1 + \exp(-s(\mathbf{W}\mathbf{f}^{(t-1)} - \mathbf{x}_0))]^{-1} \tag{2.4}$$

The weights and offsets may themselves change from cycle to cycle. There are many weight updating schemes in the literature, most of which attempt to generate

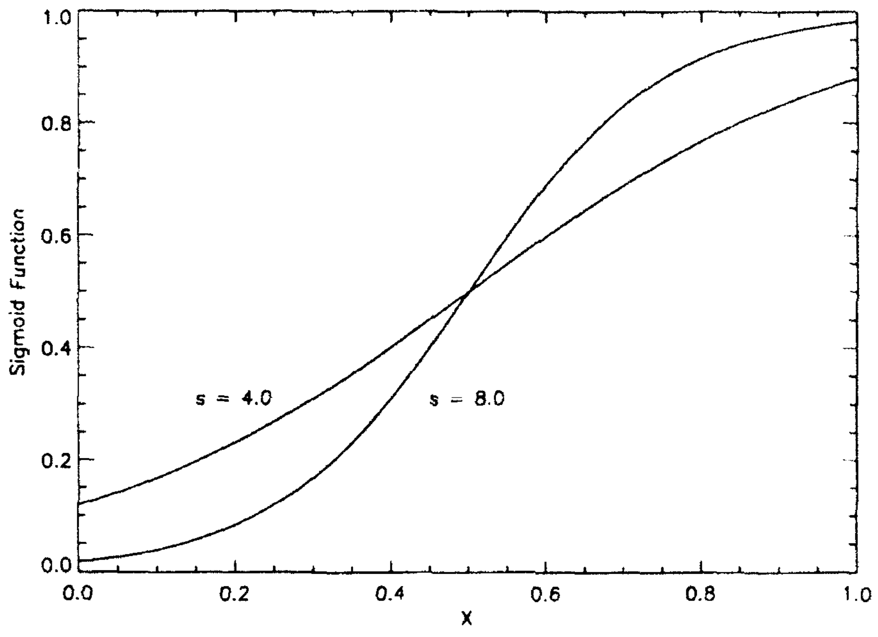

Figure 2.1

Sigmoid Functions with s = 4.0, 8.0, and $x_0 = 0.5$.

stable network states in response to given initial conditions; i.e. the vector **f** becomes fixed after a number of cycles. References 16 and 17 are collections that contain many of the important papers on weight updating algorithms. In Section 2.1 on 3-node networks and in Section 2.2 on multinode systems, the weights are not altered from cycle to cycle. However, Chapter 3 considers networks in which the weights and biases change as the node activities change.

In the next section we show how a simple 3-node system can be made to generate periodic and chaotic output.

2.1 A 3-node recurrent network

Consider the simple recurrent network of 3 nodes shown in Figure 2.2. Initially, we consider the function $y(x)$ that is generated by the group of 3 nodes in response to the input x. The output $y(x^{(t)})$ actually occurs after 2 network cycles; i.e.

$$y(x^{(t)}) = f_3^{(t+2)} \tag{2.5}$$

However, to demonstrate the property of period-doubling to chaos in a way that can be easily related to Feigenbaum's work [1], we consider the iterative generation

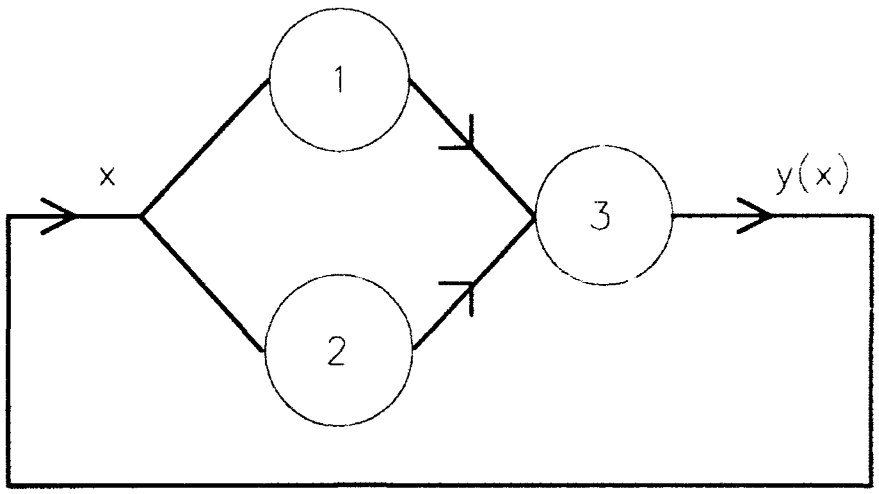

Figure 2.2

A Simple 3-Node Recurrent Network.

of the output of the 3-node group to be

$$x^{(t)} \rightarrow y(x^{(t)}) \rightarrow x^{(t+1)} \rightarrow y(x^{(t+1)}) \rightarrow \cdots \tag{2.6}$$

2.1.1 Generation of symmetric peak functions

The sigmoidal curves generated by nodes 1 and 2 of Figure 2.2 can be combined at node 3 in such a way that $y(x)$ is a symmetric peak function. This can be done by setting $W_{31} = -W_{32}$. Since nodes 1 and 2 are not connected, the weight matrix is

$$\mathbf{W} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ -\frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \tag{2.7}$$

W satisfies the normalization condition (Eq. 2.3), and the node outputs are

$$f_n = [1 + \exp(-s(x - x_{0n}))]^{-1} \qquad (n = 1, 2) \tag{2.8}$$

and

$$f_3 = [1 + \exp(-s(p(x) - x_{03}))]^{-1} = y(x) \tag{2.9}$$

where

$$p(x) = \frac{1}{2}[f_2(x; s, x_{02}) - f_1(x; s, x_{01})] \tag{2.10}$$

If the two sigmoids f_1 and f_2 have different offsets, $p(x)$ is a peak function. Figure 2.3 shows $p(x)$ for several values of the steepness parameter (s = 10, 20, 30), with $x_{01} = 0.65$ and $x_{02} = 0.60$.

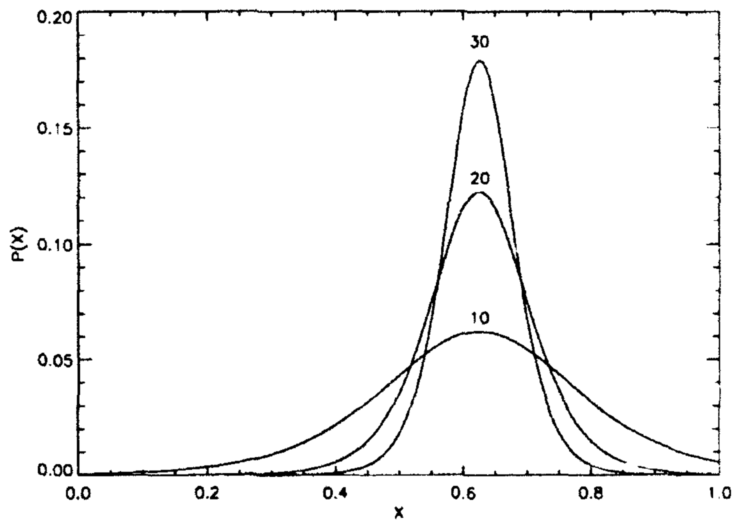

Figure 2.3

$p(x)$ with $s = 10, 20, 30$, and $x_{01} = 0.65$, $x_{02} = 0.60$.

The function $y(x)$ also depends parametrically on the offset for node 3, x_{03}. Figure 2.4 shows $y(x)$ with $x_{03} = 0.0$, and using the s ne parameters as those used in Figure 2.3. It can be shown that the asymptotic values of $y(x)$ are 0.5, and that as s increases the maximum approaches 1.0; i.e. for $x_{03} = 0.0$

$$0.5 \leq y(x) \leq 1.0 \qquad (-\infty < x < \infty) \tag{2.11}$$

If x_{03} is not zero, the baseline of $y(x)$ moves up ($x_{03} < 0$) or down ($x_{03} > 0$) the y-axis.

2.1.2 Periodicity and chaos

Figure 2.4 shows the interception of the line $y = x$ with curves of $y(x; s)$ for 3 values of the steepness parameter s.

Feigenbaum showed [1] that when the magnitude of the slope of $y(x)$ at $y = x$ is less than 1, the point of interception is a stable fixed point (x_f); i.e. iterations from all initial values of x converge to x_f. Thus, the lowest curve in Figure 2.4, with $s =$ 10, will generate iterative values of about 0.65 for x and $y(x)$, for *any* initial value of x. This can be seen from the following procedure:

1. choose any initial value on the x-axis; e.g. $x_1 = 0.2$;
2. find $y(x_1)$ on the curve;
3. move horizontally to the line $y = x$ to find x_2;

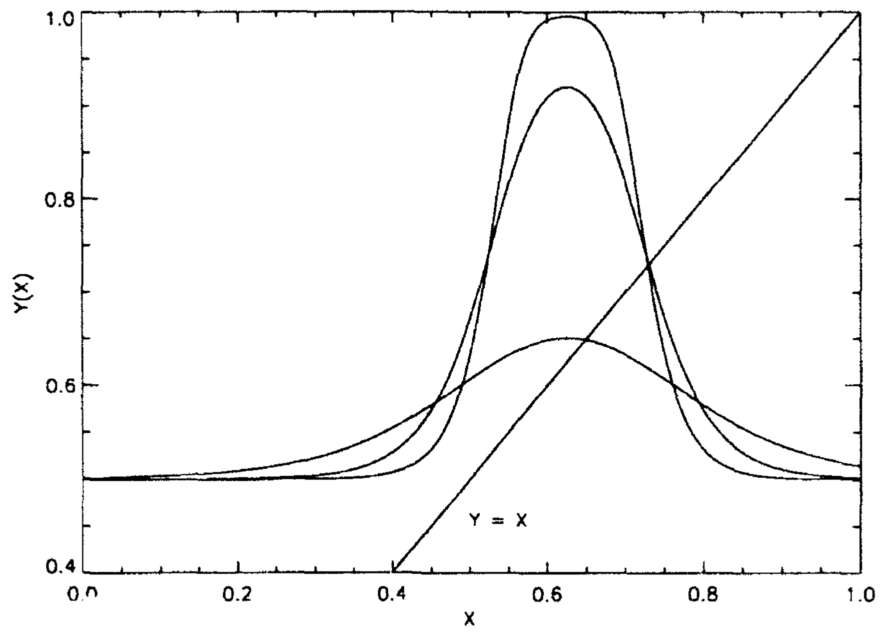

Figure 2.4

$y(x)$ with $s = 10, 20, 30$, and $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. The line $y = x$ is also shown.

4. iterate steps 2 and 3.

The iteration converges rapidly to the interception point on the curve with $s = 10$.

It is possible for a curve to have *more than one* stable fixed point. For example, consider the curve of Figure 2.5, which has been displaced to the right by setting $x_{01} = 0.8$ and $x_{02} = 0.75$, and has $s = 15.0$. In this case there are *two* stable fixed points, near 0.54 and 0.79.

In Figure 2.5 there is also an *unstable* fixed point near 0.64. At this point the slope of $y(x)$ is > 1, and iterations diverge away from the point to either of the stable points. Therefore this system can stabilize to either of two values, depending on the initial value. There are certain regions of x for which the iterative result is extremely sensitive to the initial value; e.g. near 0.64 and 0.92 in Figure 2.5.

When many sigmoidal nodes provide input to another node, the output function can contain many peaks. Therefore, large Hopfield networks can stabilize to any of many fixed patterns, and which pattern results may be very sensitive to the initial conditions.

Consider the cases in Figure 2.4 where $|y'| > 1$ at $y = x$; i.e. with $s = 20$ and 30. Now the iterations diverge away from $y = x$. This is an unstable fixed point. As s increases from 10, the iterated values settle into a cycle of period 2.

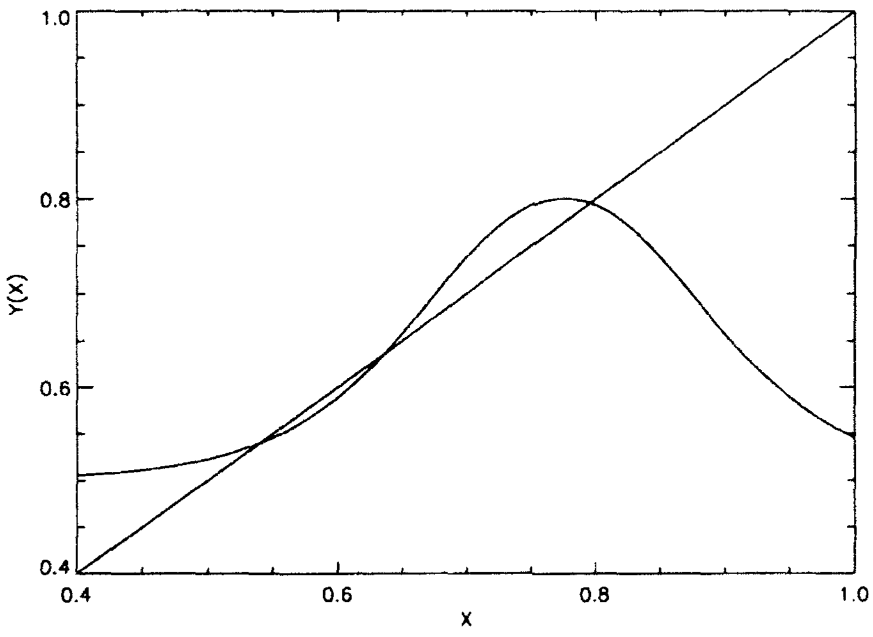

Figure 2.5

$y(x)$ with 2 stable fixed points; $s = 15$, $x_{01} = 0.8$, $x_{02} = 0.75$, $x_{03} = 0.0$. There is also an unstable point.

This *period doubling* occurs because the curve $y(y(x))$, which we write $y^{(2)}(x)$, has 2 stable fixed points. An example is shown in Figure 2.6, for $s = 15.0$.

As s increases, the two stable points of $y^{(2)}$ become unstable, 4 new stable points first occur for $y^{(4)}$ (Figure 2.7, with $s = 16.0$), then 8 for $y^{(8)}$ (Figure 2.8, $s = 16.5$) and so on. Period doubling occurs *ad infinitum*, but the spacing between successive values of s at which doubling occurs becomes geometrically smaller. Thus there is a rapid convergence to a value s_∞, beyond which the system is aperiodic, or chaotic.

Table I lists the values of s_n, found numerically, at which the n-th period doubling was first observed to occur. Table I also gives the values of Feigenbaum's constants δ_n, calculated by substituting s_n for λ_n in Eq. 1.2. This shows a rapid convergence to Feigenbaum's value of δ (4.669201..).

It is quite difficult to obtain accurate numerical estimates of s_n, for several reasons. First, for any value of s that is very near a period doubling transition, it takes hundreds of thousands of iterations of the x-values to stabilize to a precise fixed cycle. For example, Figure 2.9 shows the slow convergence of iterated x-values when $s = 12.9$, near the first period doubling. As s approaches s_1 (12.9735..) the convergence gets slower, and as s increases further a stable oscillator of period 2 eventually emerges, as shown in Figure 2.10 for $s = 13.1$.

The closer s is to a period-doubling transition, the greater is the number of itera-

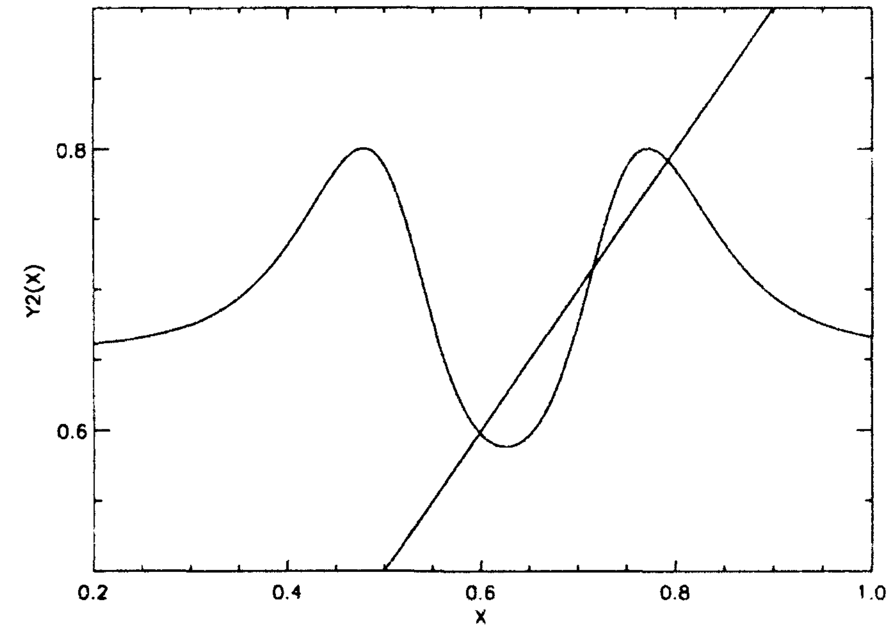

Figure 2.6

$y^{(2)}(x)$ with 2 stable fixed points; $s = 15$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. There is also an unstable point.

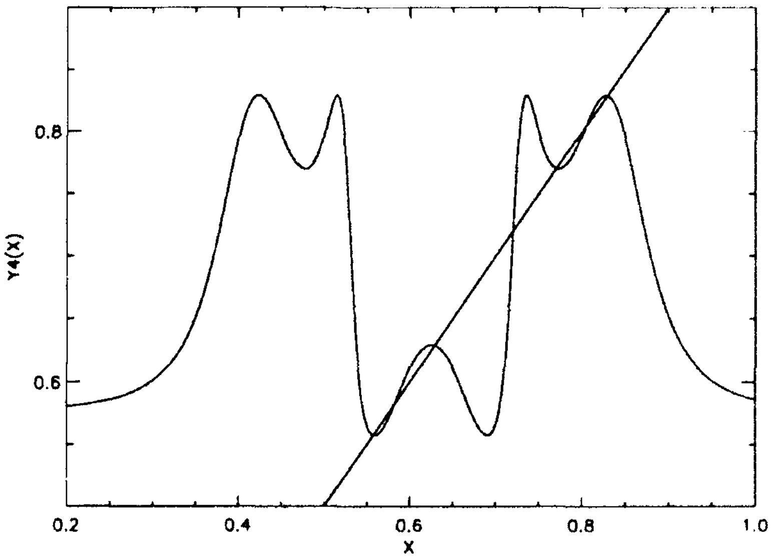

Figure 2.7

$y^{(4)}(x)$ with 4 stable fixed points; $s = 16$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. There are also 3 unstable points.

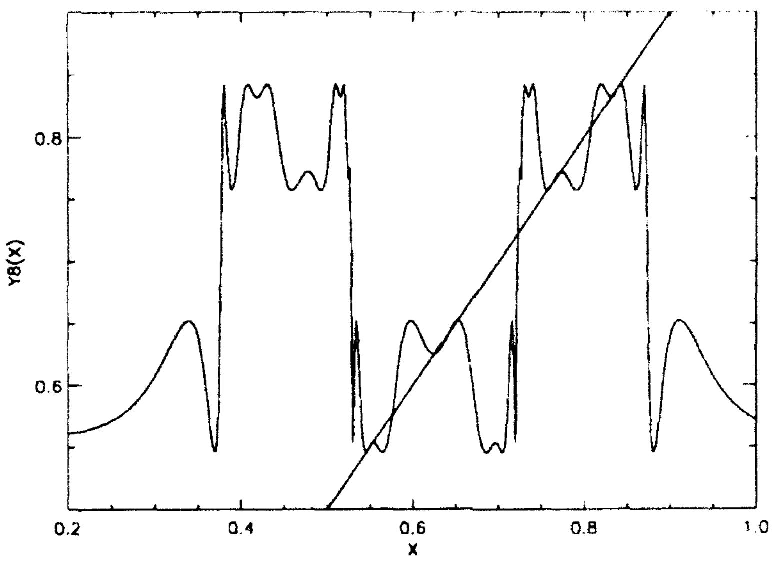

Figure 2.8

$y^{(8)}(x)$ with 8 stable fixed points; $s = 16.5$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. There are also 7 unstable points.

n	s_n	period	δ_n
1	12.9735	2	2.503
2	15.4103	4	4.124
3	16.3838	8	4.579
4	16.61984	16	4.646
5	16.6713832	32	4.665
6	16.6824763	64	4.669
7	16.6848542	128	-
8	16.68536348	256	-

Table I

Values of s at which period doubling first occurs

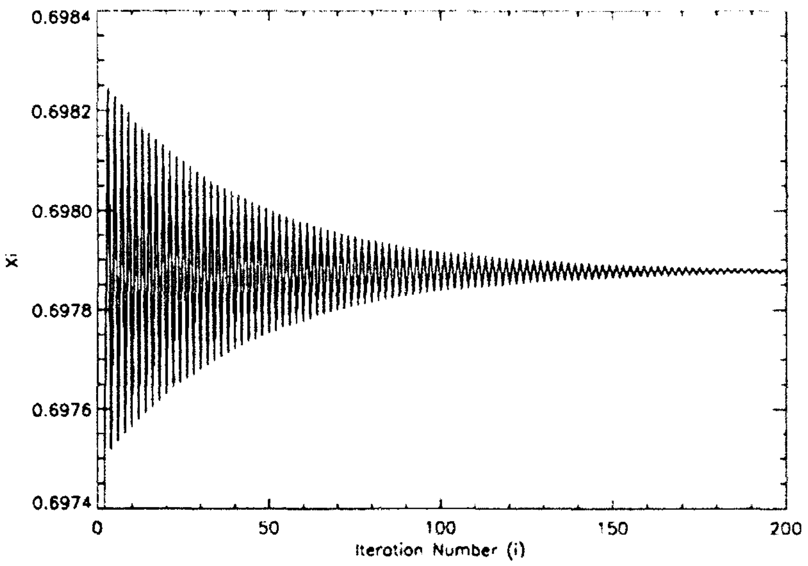

Figure 2.9

Slow convergence of x_i when $s = 12.9$, near the first period doubling

.

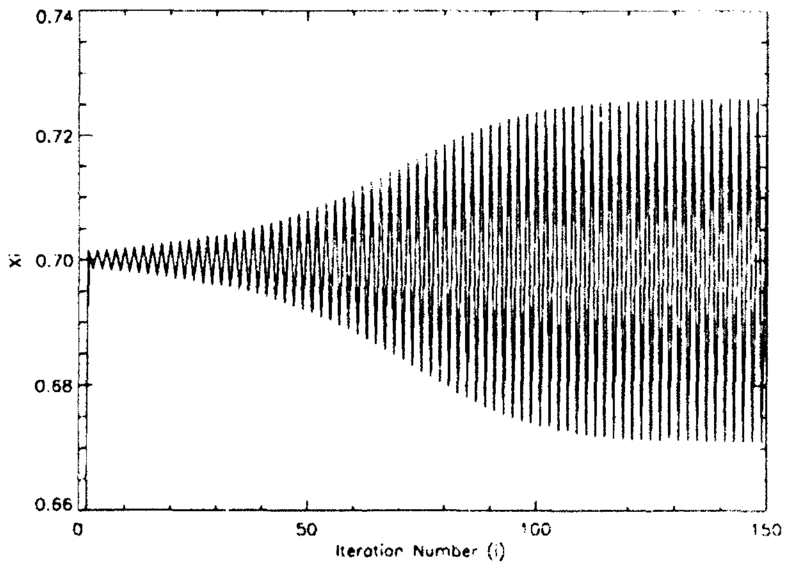

Figure 2.10

Convergence to a cycle of period 2 ($s = 13.1$).

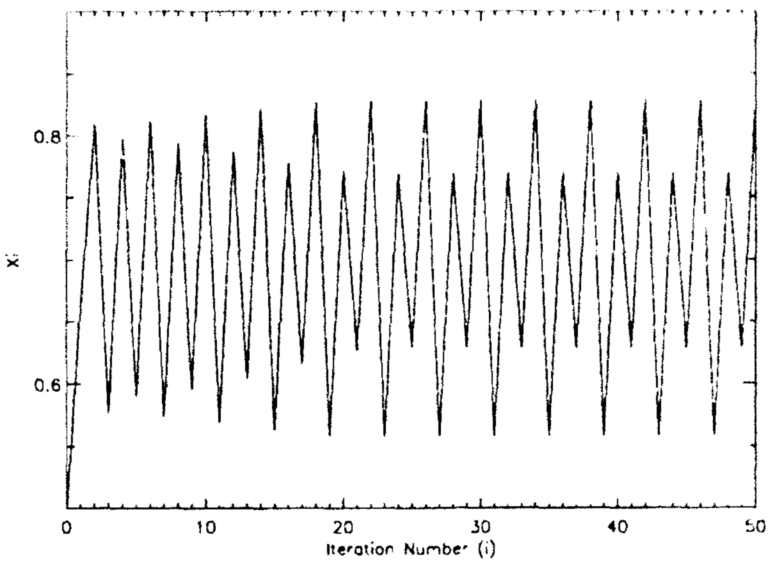

Figure 2.11

Cycle of period 4 ($s = 16.0$).

tions required to determine the stable cycle, and so some criteria must be chosen to define the point at which period doubling is accepted to have occurred. Our algorithm performs the following two tests for a cycle of period N:

$$|x_i - x_{i-N}| < \Delta \tag{2.12}$$

and

$$|x_i - x_{i-M}| > \epsilon \qquad (M < N) \tag{2.13}$$

We used $\Delta = 1.0\text{E-}12$ and $\epsilon = 1.0\text{E-}08$. The second condition, Eq. 2.13, establishes the uniqueness of at least one other member of a cycle of period N. After every iteration, the new point must be checked against up to N previous points. Thus, if for example the period = 1024, thousands of checks must be performed on hundreds of thousands of iterations. This is quite CPU intensive, taking several minutes on a typical Unix workstation. We use a Hewlett Packard 9000/720. All calculations were done in double precision, using a C program.

Figures 2.11 and 2.12 show example of cycles of periods 4 and 8 respectively. They show that when the value of s is *not* near a transition, a stable cycle is quickly established after only a few iterations.

Table I shows values of s that generate period doublings up to 256. Cycles of period 512 and 1024 were also observed with $s = 16.6854726$ and 16.6854960 respectively, but no accurate estimates of s_9 and s_{10} were attempted numerically because the differences are approaching the limit of double precision accuracy.

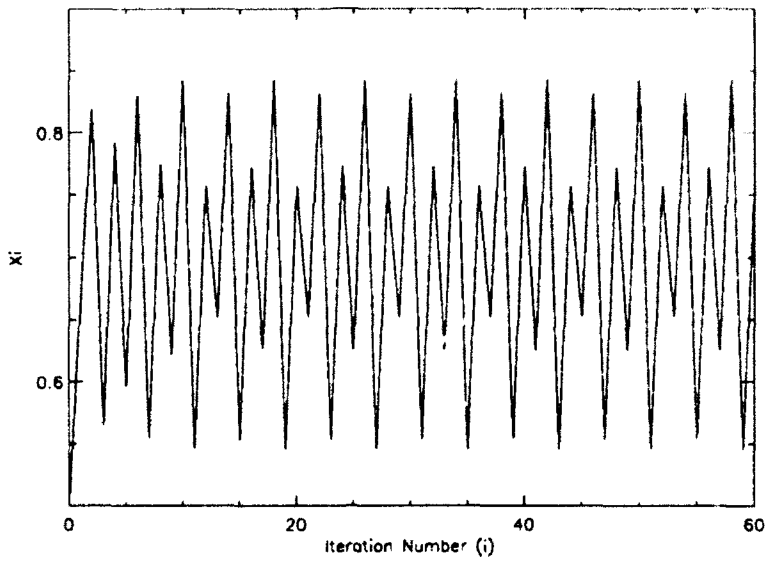

Figure 2.12

Cycle of period 8 ($s = 16.5$).

Table I also indicates that a region of chaotic (aperiodic) signal generation would be expected for values of $s > 16.7$, and in fact no periodicity was observed immediately beyond this value. Figure 2.13 shows an example of chaotic iterative x values for $s =$ 17.0. The values shown are an arbitrary set of 200 consecutive iterations.

Iterative periodicity and chaos can be represented as trajectories in xy-space. If we take each input signal x_i and its output y_i (i.e. x_{i+1}) to be a coordinate pair, then the movement of these points represents a trajectory. Figure 2.14 shows the trajectory for the cycle of period 8 that was shown in Figure 2.12. Figure 2.14 is actually a plot of 101 points from the 400th to 500th iterations. This shows that the system is constrained to move in a closed 8-point trajectory. Each of the 8 points must, of course, lie on the curve $y(x)$. Figure 2.15 is the trajectory for the chaotic system shown in Figure 2.13. In this case, the trajectory is *not* closed. Plotting more points results in completely filling the curve $y(x)$ to which the trajectory is confined.

As s increases, the system remains chaotic until a cycle of period 7 appears near $s = 20.0$. Period doubling then reoccurs as s increases, as summarized in Table II. The way in which this region of periodicity emerges from a chaotic region can be understood by looking at the curve $y^{(7)}(x)$ for $s = 20.0$, where the period is 7. Figure 2.16 shows that this curve has 7 points where $|y'| < 1$, where the line $y = x$ intercepts maxima or minima. Figure 2.17 shows the same curve, but with $s = 19.0$, just before the onset of periodicity.

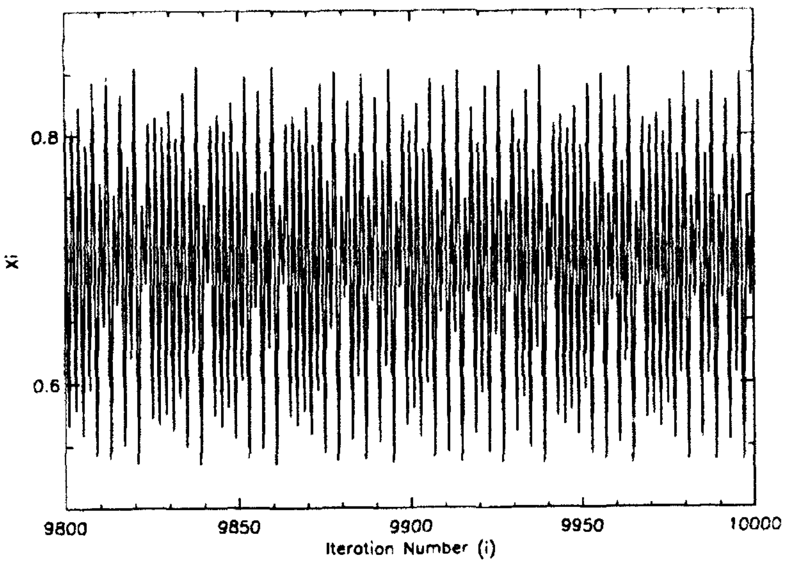

Figure 2.13

Chaotic signal generation ($s = 17.0$).

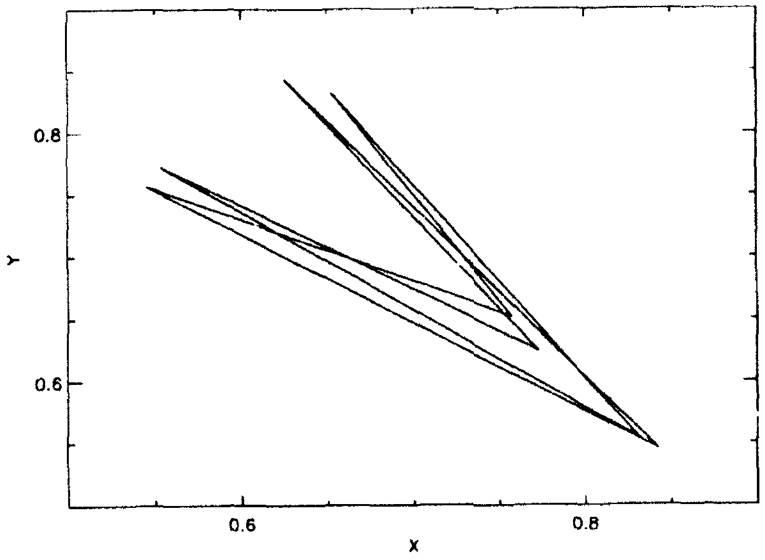

Figure 2.14

Trajectory of a cycle of period 8 ($s = 16.5$).

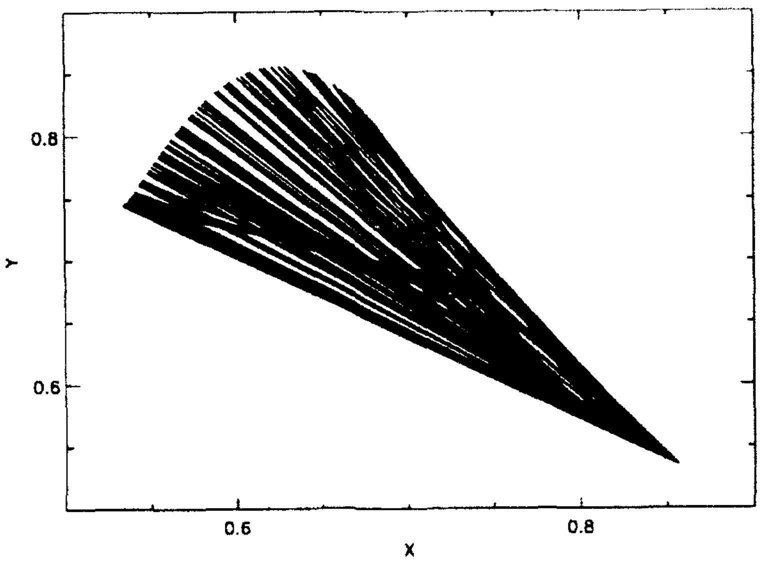

Figure 2.15

Chaotic trajectory ($s = 17.0$).

n	s_n	period
0	19.996	7
1	20.090	14
2	20.131	28
3	20.140	56
4	20.1415	112
5	20.1419	224

Table II

Approximate values of s in second period doubling region

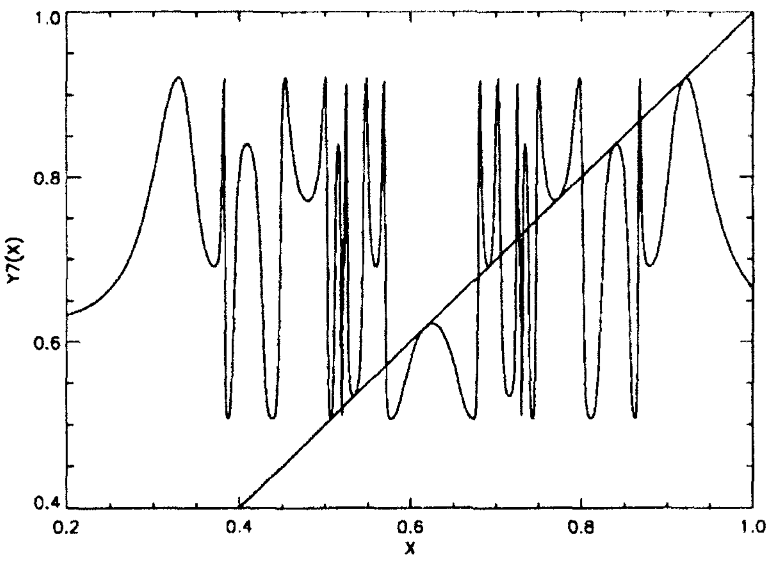

Figure 2.16

$y^{(7)}(x)$ with 7 stable fixed points; $s = 20.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

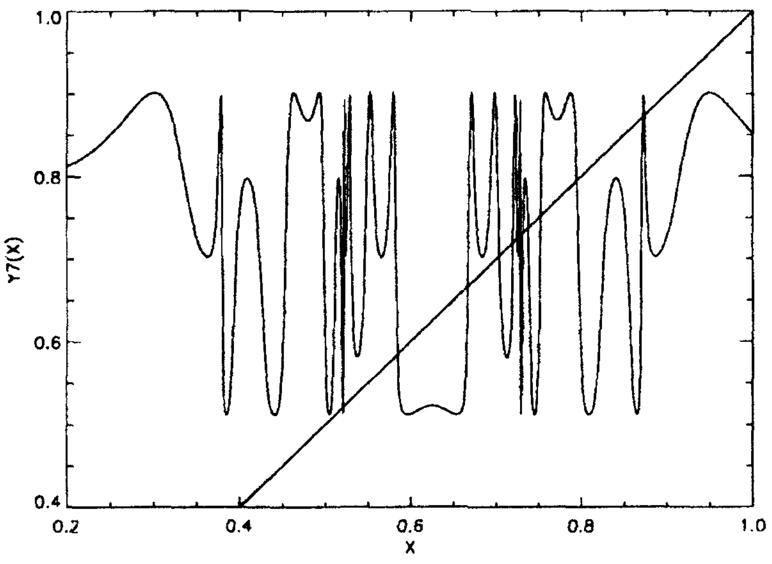

Figure 2.17

$y^{(7)}(x)$ with no stable fixed points; $s = 19.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

n	s_n	period
0	20.600	9
1	20.602	18
2	20.6032	36
3	20.60351	72
4	20.60358	144
0	20.95	8
1	21.00	16
2	21.024	32
3	21.029	64
4	21.030	128

Table III

Approximate values of s in 3rd and 4th period doubling regions

The second region of periodicity spans a much smaller range of s than the first, and at $s = 20.15$ the system is again chaotic. However, because of the emergence of new maxima and minima in the curves $y^{(n)}(x)$, further regions of period doubling occur. At $s = 20.60$, a cycle of period 9 is established, period doubling reoccurs, then again at $s = 20.95$ a cycle of period 8 appears. These 3rd and 4th regions of period doubling are summarized in Table III.

Beyond the 4th period doubling region the system is chaotic until about $s = 24$, where a cycle of period 3 appears. Now the sigmoids are so steep that they are almost step functions. At $s = 23.0$ (Figure 2.18) $y^{(3)}(x)$ has no value for which $|y'| < 1$ where $y = x$, but when $s = 25.0$ (Figure 2.19) there are three stable fixed points. This system of period 3 remains stable as s increases from 24 to 47; i.e. no period doubling occurs. This is due to the flat extrema produced by the quasi-step functions.

At $s = 48$, a cycle of period 6 occurs. However, this is not a process of period doubling leading to chaos, but rather a series of abrupt and unpredictable transitions. As s increases, cycles of period 5, 13, 4, 6, 5 and 7 appear for $s = 50, 51, 52, 61, 62$ and 70 respectively. Figures 2.20 and 2.21, showing $y^{(3)}(x)$ for $s = 47$ and 49 respectively, demonstrate the origin of these sharp transitions, arising near discontinuities in the steep quasi-step functions. Figure 2.22 shows the 6 fixed points for $s = 49$, and Figures 2.23 to 2.25 are examples of cycles with periods 5, 13 and 4. In this domain of s the fixed points jump erratically from one flat region of a y iterate to another.

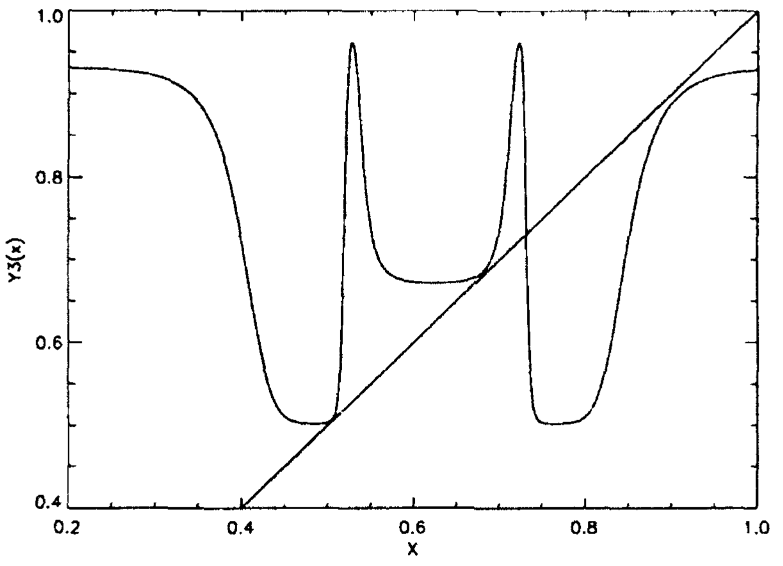

Figure 2.18

$y^{(3)}(x)$ with no stable fixed points; $s = 23.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

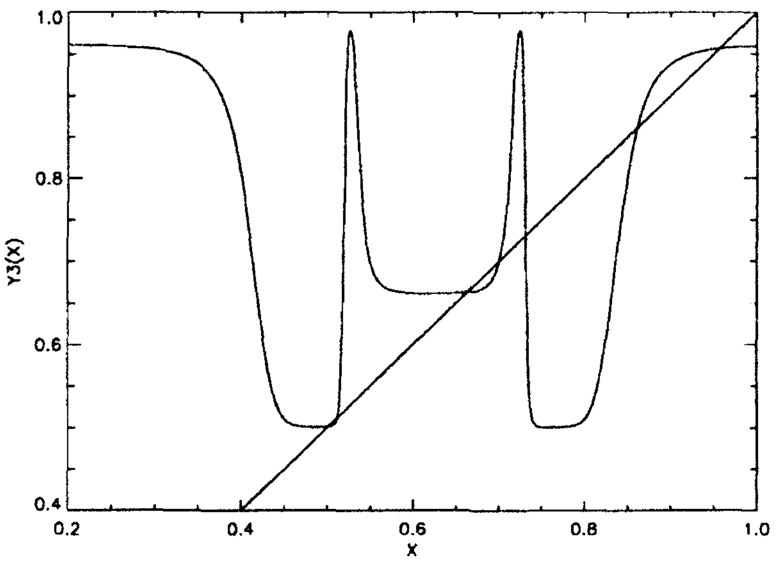

Figure 2.19

$y^{(3)}(x)$ with 3 stable fixed points; $s = 25.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

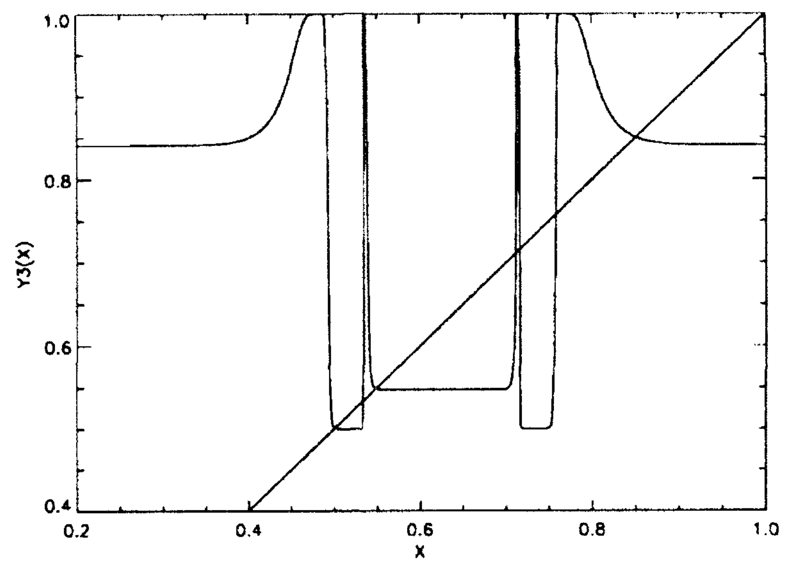

Figure 2.20

$y^{(3)}(x)$ with 3 stable fixed points; $s = 47.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

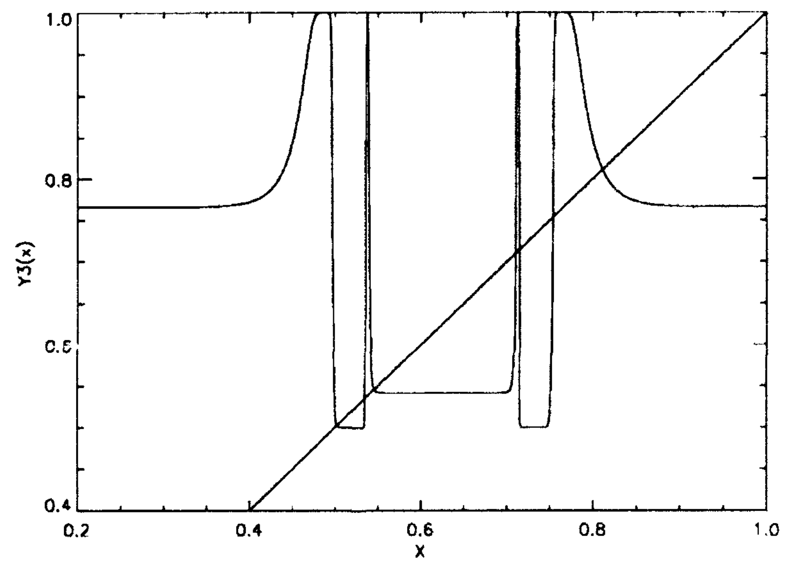

Figure 2.21

$y^{(3)}(x)$ with no stable fixed points; $s = 49.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

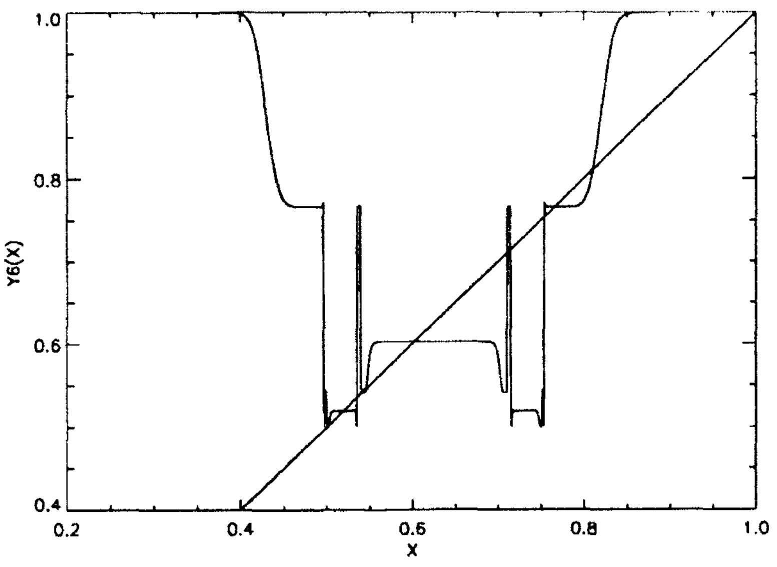

Figure 2.22

$y^{(6)}(x)$ with 6 stable fixed points; $s = 49.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

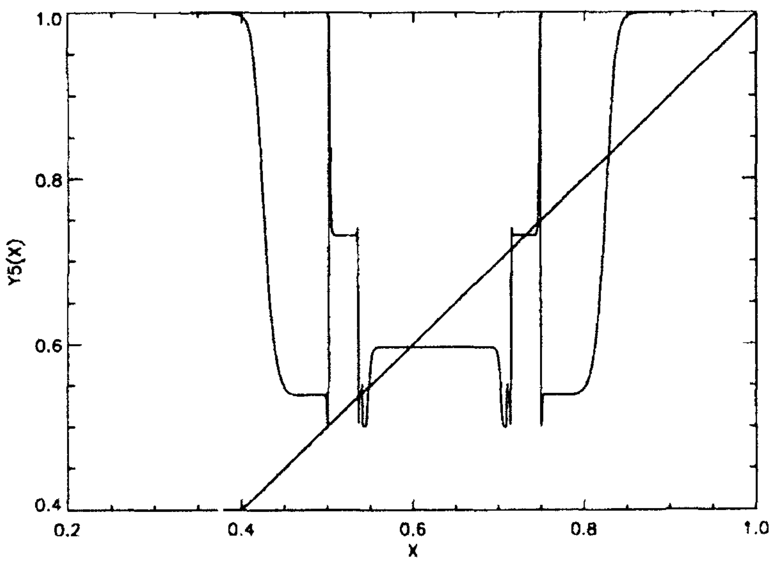

Figure 2.23

$y^{(5)}(x)$ with 5 stable fixed points; $s = 50.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

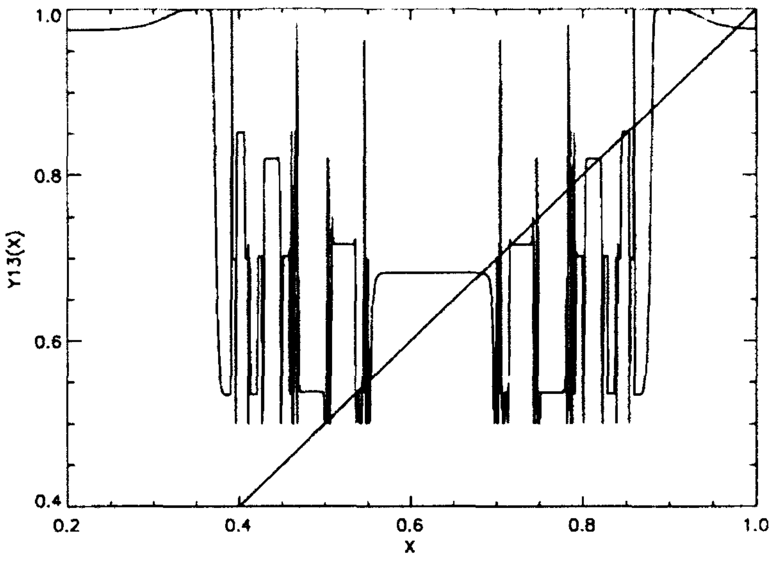

Figure 2.24

$y^{(13)}(x)$ with 13 stable fixed points; $s = 51.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

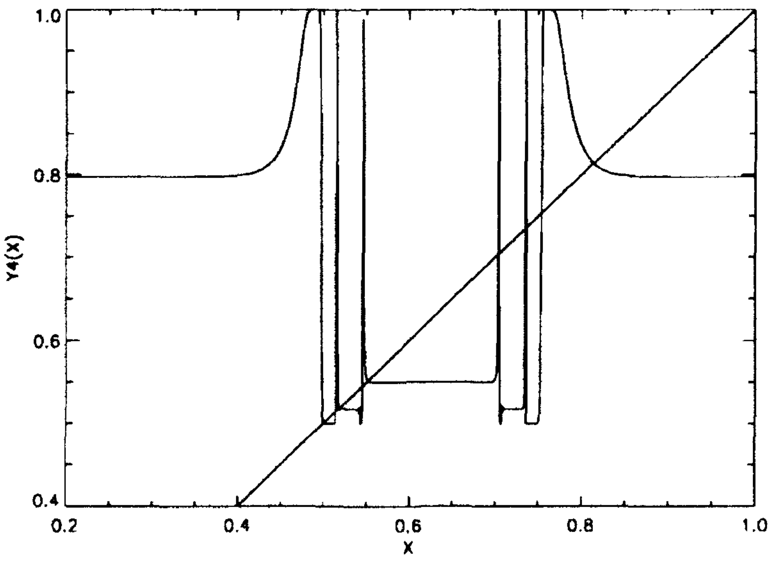

Figure 2.25

$y^{(4)}(x)$ with 4 stable fixed points; $s = 60.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

2.2 Multinode Networks

Having determined that in a very simple recurrent network the generation of stable, periodic or chaotic signals can be controlled by a single global parameter, we now determine if stability can be controlled similarly in large ANNs. The simple mathematical analysis of the iterated functions that was possible for the 3-node system is no longer possible when the output signal from any given node is fed back through many other nodes whose outputs may not reach the original node until many iterations are completed; i.e. the signal that is fed back is strongly coupled to many previous iterations. We have therefore focussed on the numerical results of iteratively calculating the vector of node activities in large networks.

First we look at nets in which the weights and offsets are *fixed* at some random initial values (sections 2.2.1 and 2.2.2) and the system is iterated with various values of the node steepness parameter s. These rigid systems do not have adaptive or learning properties, although it may be possible to use them to store and recall different patterns depending on the initial input vector, and their stability properties are relevant to an understanding of large nets with plastic parameters. Section 2.2.1 considers completely connected ANNs.

Biological systems do not have completely connected sets of neurons. It would be physically impossible to connect all the billions of neurons in a mammal, for example. Similarly, in a simulation of a recurrently connected ANN, computer processing speed and storage capacity limit the number of connections per node. In section 2.2.2 we describe the stability of a partially connected ANN with fixed parameters.

In Chapter 3 we consider *dynamic* recurrent networks, in which the weights and offsets change from cycle to cycle, depending on the input and output at each node. Chapter 3 concludes by describing the interaction of a dynamic ANN with a simulated environment, through nodes receiving external input and transmitting motor output.

2.2.1 Completely connected nets with fixed weights and offsets

We have simulated several completely connected multinode networks. The nodes had no self-feedback ($W_{nn} = 0$). The weights for each node were chosen randomly between -1 and +1, and then normalized (Equation 2.3), with the normalization constant generally chosen to be the number of nodes in the network. The only purpose of this normalization was to ensure that each node could receive roughly the same total input signal. The offsets, contained in the vector $\mathbf{x}_0$, were chosen randomly between 0 and 1.

The state at time t of a recurrent network of N nodes is given by the vector $\mathbf{f}^{(t)}$. It is calculated from the state at the previous cycle using Equation 2.4. The initial components of the state vector for each network were chosen randomly between 0 and 1.

When a recurrent network operates iteratively, periodicity must be detected by testing all the components of the state vector with all the components from previous cycles (cf. Equations 2.12 and 2.13). For example, for a 100 node network, the outputs from all the 100 nodes must be identical to those from a previous cycle for the system to be termed periodic.

From the work described in the previous section on 3-node networks, we expected that periodicity and chaos should occur for some random multinode networks, and that it should be possible to control this behaviour by adjusting the node steepness parameter s for the whole network.

Initial work on nets containing 20 and 100 nodes supports this. A 20-node network showed cycles of period 1, 7, 14 as s increased from 1.0 to 2.1. However, beyond about 2.194, we could detect no periodicity. The 20-node system then appears to be chaotic. This aperiodicity persists until s reaches about 6.0, where a cycle of 1 occurs. For higher values of s, erratic periodicity occurs, much as in the 3-node case, with cycles of period 38, 1, 142, 1, 31, 33, 13,.. being observed in the region $s = 10$ to 20.

A particular randomly initiated 100-node network was stable for $s < 0.69$, i.e. it converged to a cycle of period 1. However, above this value we detected no periodicity. The transition from stable behaviour to chaos seems to be much more abrupt for this larger network. This is not to say that *all* randomly initiated networks will exhibit a transition to chaos when s is increased. In fact, most of these networks simply iterate to a stable state. However, those that do exhibit a transition to chaos can all be stabilized by reducing s.

2.2.2 Partially connected nets with fixed weights and offsets

Because it was a convenient structure for the subsequent studies described in Chapter 3, we focussed on a network of 90 nodes, each with input connections from 10 other different nodes. The connections were chosen randomly from anywhere in the network. Other network sizes and numbers of connections have also been studied, but the results are essentially the same; i.e. the stability can be controlled by varying s.

The connection weights in the 90-node system were chosen randomly between 0.1 and 0.9, and the sign was also assigned randomly. The offsets were chosen randomly between -1.0 and 1.0.

The results were quite straightforward. For $s \leq 5.6$ the network converged to a fixed state, and there was a sharp transition to chaos around $s \geq 5.7$. We observed no region of period doubling in this system. However, other randomly initialized systems may show periodicity. Since our goal is to develop adaptive, learning machines, a detailed study of the conditions under which period doubling may occur in the transition to chaos was not performed.

3. Dynamic Recurrent Networks

A *dynamic* recurrent network is one in which the weights and offsets change from cycle to cycle, depending on the input and output at each node.

The dynamic variation of connection weights and offsets is similar to the changing synaptic strengths and thresholds in biological neural systems, and offers the possibility of machines that are capable of learning from, remembering, and adapting to their environment. This requires some dynamic interaction between the network and the environment through nodes that receive sensory input and output nodes that drive motor responses.

The structure of such a dynamic recurrent ANN is described in section 3.1, together with its properties in the *absence* of external input. Output node signals are used to drive a simulated machine and its two-dimensional trajectories are shown to depend on the global node steepness parameter s.

In section 3.2 we describe the effect of sensory input on the movement of the simulated machine.

3.1 A partially connected dynamic recurrent ANN

The structure of the recurrent ANN described in this section was intended to provide the basis for studies of the dynamic interaction of an ANN with a simulated environment containing an emitting source, results from which are given in section 3.2.

Our dynamic recurrent ANN (Figure 3.1) has 3 distinct regions, which we call input, internal, and output regions. Sensors only have connections to the input region, and motor nodes are only connected to the output region. Nodes in all 3 regions are partially interconnected (randomly), so that the processing is recurrent throughout the ANN. To summarize, the following functions are performed in each region:

1. **input region:** signals from sensors and other network nodes are received, processed and passed to other nodes in the network by a limited number of randomly chosen connections;

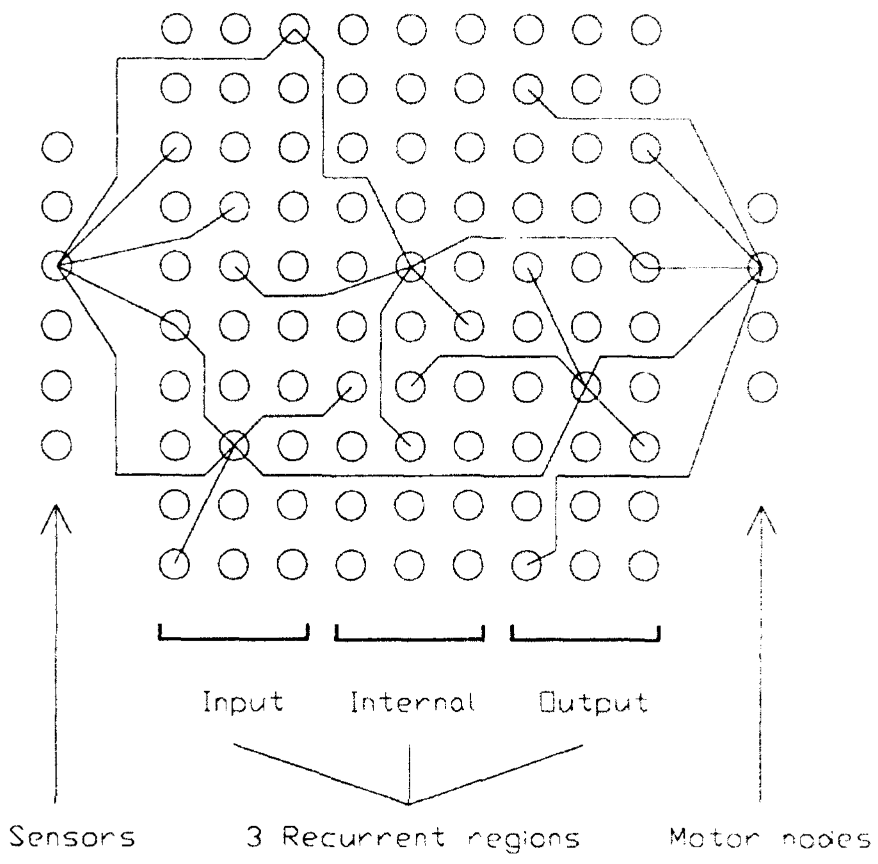

Figure 3.1

A 90-node dynamic recurrent ANN with three processing regions.

2. **internal region:** signals are received from and passed to randomly connected nodes; the nodes in this region have no direct external input or output; and

3. **output region:** signals are received from and passed to anywhere in the network, but not directly from the input sensors; the motor nodes are randomly connected to a limited number of the nodes in this region.

An example of this arrangement is shown schematically in Figure 3.1. In each of the regions, a node is shown with 5 of its possible connections. The sensors only provide output signals, and the motor nodes only receive input. The recurrent nodes may be connected for either input or output, or both. In our simulation, it is only necessary to store the input connections for each node since the activity of any node can be calculated from its input at each cycle. The node activities are stored from the previous cycle and then updated after all the new activities have been calculated. Each node is initialized to have the same number of randomly chosen input connections, and every node provides output to at least 2 other nodes.

We used 90 recurrent nodes in all the simulations described subsequently. The nodes were divided into 30 per region, as shown in Figure 3.1. This was a convenient number for reasonable CPU times, and provided enough nodes in each recurrent region for at least 10 unique input connections per node without it being difficult to randomly assign the connections. The sensors each had 10 random connections to different nodes in the input region, and each motor node had 10 random connections from different nodes in the output region. The connectivity of our network is thus about 10%. We have not yet studied the effect of changing the connectivity.

In this section, the sensor outputs were all fixed at zero. The weights for the connections from the sensor nodes to the input region were all set to 1.0. This is unimportant here because the sensor outputs were always zero. The form of the sensor response as a function of incident intensity is given in section 3.2.

The weights for the recurrent node connections were randomly initialized between 0.1 and 0.9, and the signs were also assigned randomly. The offsets were randomly initialized between -1.0 and 1.0. The weights and offsets of the recurrent nodes were changed dynamically, as described in section 3.1.1, after every iterative calculation of the network state; i.e. the node activity vector.

The motor nodes were connected with constant weights of magnitude 1.0 to randomly chosen recurrent nodes in the output region. The sign was alternated between the random connections; i.e. these weights were (1.0, -1.0, 1.0, -1.0,..). The motor node weights did not change during the simulation. However, their offsets did change dynamically (see section 3.1.1).

3.1.1 Dynamic variation of the weights and offsets

For a recurrent node i at cycle t, which received an input f_j^{t-1}, from the jth node and generated an output f_i^t, the change in the weight connecting nodes i and j that we have used is:

$$dW_{ij} = \alpha \frac{W_{ij}}{|W_{ij}|} f_j^{t-1} f_i^t - \gamma W_{ij} \qquad (3.1)$$

This is a modified form of the synaptic strength modification rule proposed by Hebb [19]. The principal effect is to strengthen connections carrying strong input signals that contribute to strong output signals. Because our weights may be positive or negative, we have included a sign term to ensure that both types can increase in magnitude, and we include a decay term (γ) so that the weight magnitudes are bounded. The parameter α controls the maximum change that can occur at each cycle; i.e. it is a growth rate.

We also allow the offsets of the recurrent and motor nodes to vary dynamically in the folowing way:

$$dx_{0i} = \beta(f_i^t - 0.5) \tag{3.2}$$

This ensures that every node continually adjusts its offset so that the node output approaches the centre of its range. No node can be permanently *off* because its offset is too high, or permanently *on* because its offset is too low. This technique has the following consequence: in a recurrent system that is approaching stability *all the node outputs will approach 0.5.* However, for networks that receive changing sensory input, or are operating in the chaotic regime, the offsets change continually.

From Eq. 3.1 it can be seen that for a system approaching stability, the weight change approaches zero and

$$|W_{ij}| \rightarrow \frac{\alpha}{4\gamma} \tag{3.3}$$

If, for example, the growth rate is set to ten times the decay rate, then all the recurrent node weights will approach a magnitude of 2.5. Reasonable values of the update parameters are:

$\alpha = 0.01$;

$\gamma = 0.001$; and

$\beta = 0.01$.

These values ensure that the maximum change in a weight is 0.01 and in and offset it is 0.005 at any iteration; i.e. normally less than 1% change for weights and offsets.

3.1.2 2D movement of a simulated machine

Four motor nodes are shown in Fig. 3.1. Since their outputs are all positive, we used the output of two of them to determine a change in the x-coordinate, one node output giving $+\delta x$, the other $-\delta x$. Similarly, the other two motor nodes determine the change in y. The magnitude of the (x, y) steps taken at each cycle may be scaled for convenience. Consequently, the units on the trajectory diagrams shown in this Chapter are arbitrary.

Now we look at the trajectories generated for different values of the recurrent node steepness parameter s_r. The motor nodes are also sigmoidal in our system, and the steepness parameter (s_o) for all of them was assigned a fixed value of 4.0. For each motor node, this generates a roughly linear response to the total input that it receives, at least in the region around the offset.

Figure 3.2 is an example of the trajectory of a system that is converging. To generate this curve, the initial random network was iterated for 5,000 cycles to allow

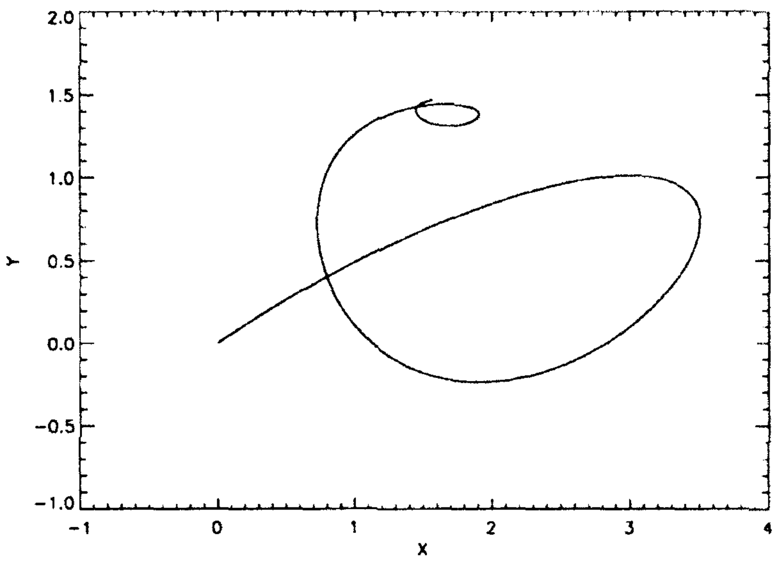

Figure 3.2

Convergent trajectory with $s_r = 0.4$ and no sensory input

the weight and offset schemes to take effect, using the update rate parameters given in section 3.1.1. Then the motion of the system was plotted from the (0,0) coordinate position for a further 10,000 iterations. The movement along the curve then represents the change in (x, y) for each iteration. The updating scheme clearly generates a smooth convergence.

The trajectories spiral to fixed points for low values of s_r, and become chaotic as s_r increases. As in the case of large systems with fixed weights and offsets (section 2.2), there is a sharp transition from systems that converge to those that become aperiodic. In our system, this occured between $s_r = 0.49$ and 0.50.

At first sight, the trajectories of systems that have $s_r \geq 0.5$ appear to be *closed*, even though the node activity vector was determined to have no exact periodicity. With $s_r = 0.5$ for example, Fig. 3.3 shows the trajectory generated by 10K iterations after an initial 500K iterations had been performed from the random initial state to ensure that it had not in fact converged to a periodic system. Figure 3.3 certainly appears to be a periodic orbit. However, when any section of the trajectory is greatly magnified, it is clear that the path never exactly repeats itself. It is chaotic. Figure 3.4 is a magnified view of the top right hand corner of the orbit shown in Fig. 3.3. If the number of iterations is increased, the lines in Fig, 3.4 simply become more closely spaced; e.g. the same region of the trajectory is shown in Fig. 3.5 after 20K iterations.

As s_r increases, the quasi-periodic trajectory first becomes more complex (Fig. 3.6), then less confined (Fig. 3.7), and finally completely unstructured (Fig. 3.8).

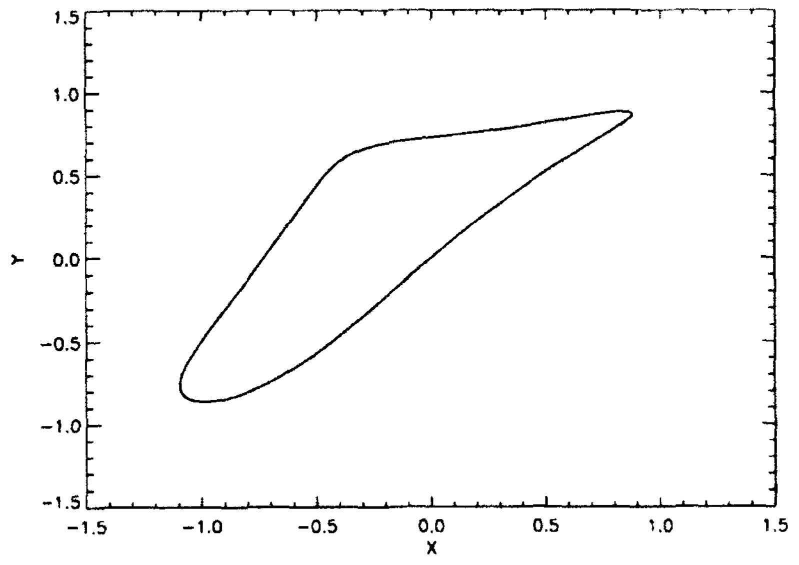

Figure 3.3

Quasi-periodic trajectory with $s_r = 0.5$ and no sensory input

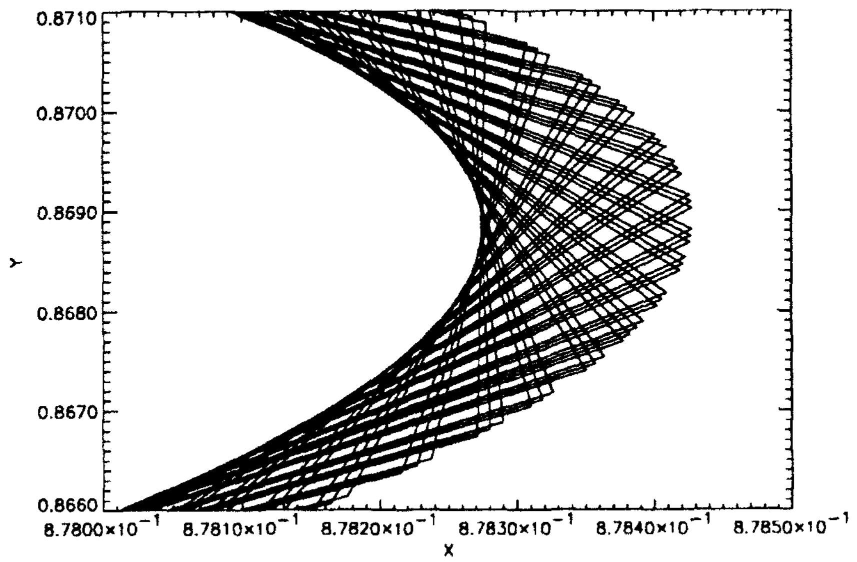

Figure 3.4

Magnified view of the chaotic trajectory with $s_r = 0.5$ after 10,000 iterations

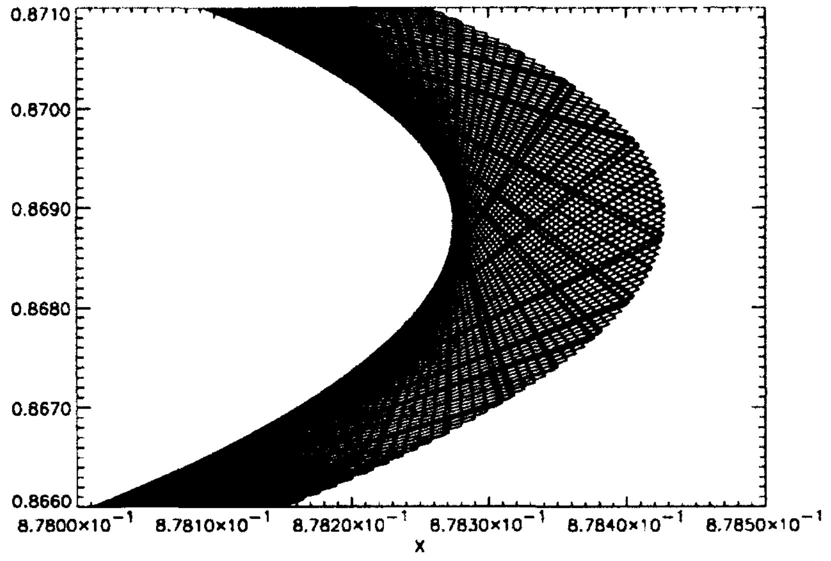

Figure 3.5

Trajectory with $s_r = 0.5$ after 20,000 iterations

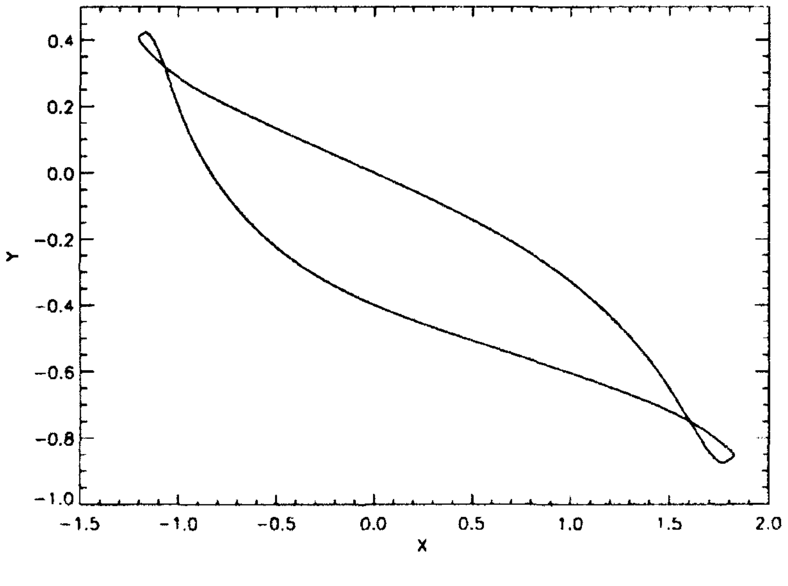

Figure 3.6

Quasi-periodic trajectory with $s_r = 0.6$ and no sensory input

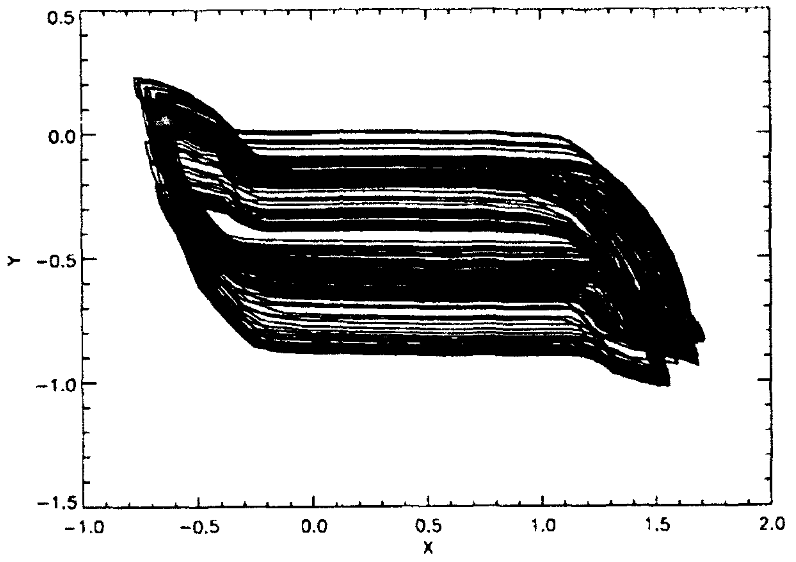

Figure 3.7

Trajectory with $s_r = 0.8$ (10,000 iterations)

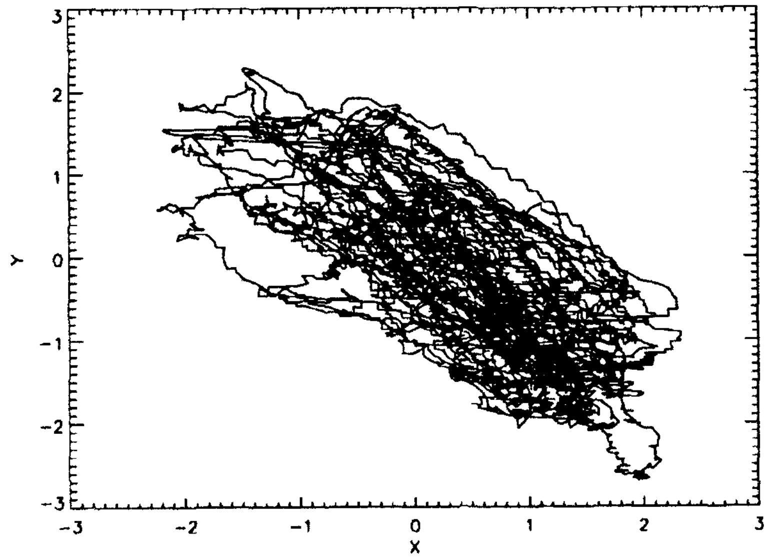

Figure 3.8

Unstructured trajectory with $s_r = 1.0$ (10,000 iterations)

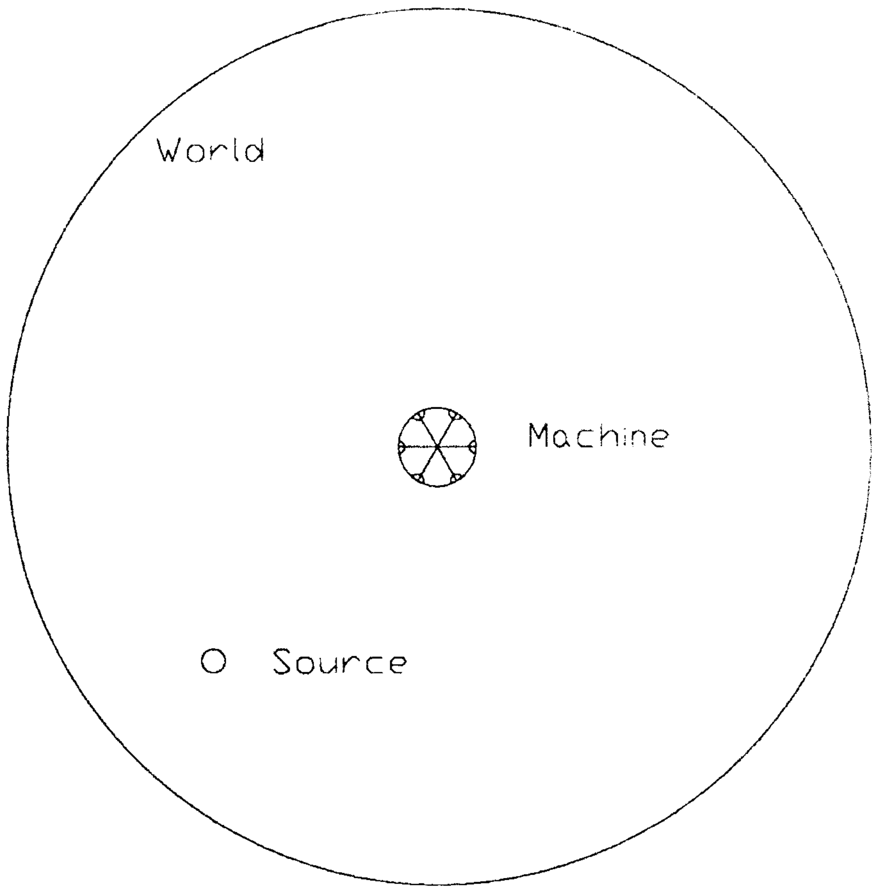

Figure 3.9

Simulated machine in a 2D world with an emitting source

3.2 Effect of Sensory Input

We have simulated the movement in two dimensions of a circular machine that is controlled by the recurrent neural network shown in Fig. 3.1. The six sensors are equally spaced around the circular machine, which is confined to move in a circular world containing an emitting source. The arrangement is shown in Fig. 3.9.

The dimensions and units used in the simulation are arbitrary because the motor node outputs can be scaled to adjust the machine step size, and the source intensity can be varied at will. We normally use a world radius of 10 and a machine diameter of 1. The intensity at the source is normally in the range 1 to 10, and is modelled to decrease as the inverse of the distance from the source. The angular size of the sensor and the angle that the sensor surface makes with a line joining the source to the sensor centre are both included in a realistic estimate of the intensity at each sensor. Also the position and size of each sensor is used to determine which sensors are partially or totally illuminated. The response of a sensor that is not illuminated

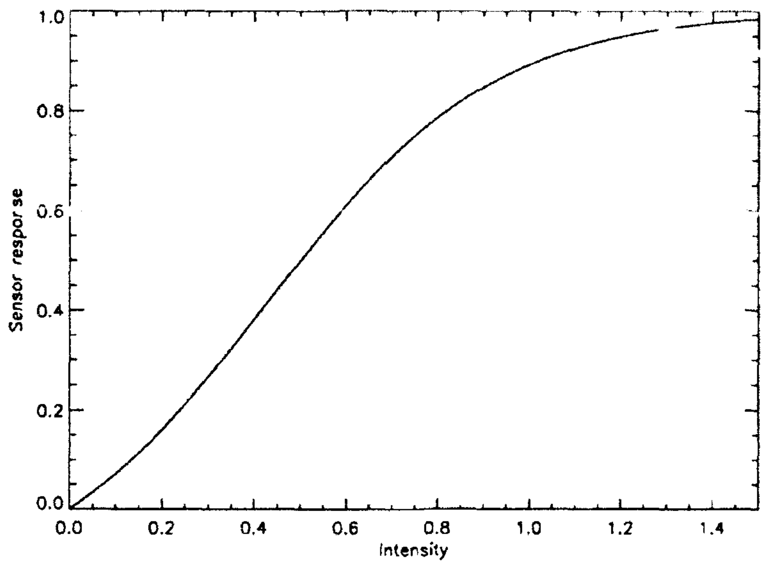

Figure 3.10

Sensor response as a function of incident intensity

at all is zero.

The function that we have used to generate the sensor response to incident intensity is shown in Fig. 3.10. It has a roughly linear response in the centre of its dynamic range, and saturates at high intensity. The equation used to generate this response is:

$$f(x) = \frac{1 - g(x)}{1 + \nu g(x)} \tag{3.4}$$

where

$$g(x) = \exp(-s_i x) \tag{3.5}$$

and

$$\nu = \exp(s_i x_{0i}) \tag{3.6}$$

s_i and x_{0i} define the steepness and offset respectively of the curve shown in Fig. 3.10. The subscript i is used to indicate *input* to the network. $s_i = 4.0$ gives a roughly linear response near x_{0i}. The value of x_{0i} was chosen to give $f(0.5) = 0.5$.

Thus, as the machine moves about the world, driven by the activity of the recurrent nodes, the sensor responses change dynamically and are fed into the network. This provides a continual feedback from the environment. However, it should be noted that the sensor input to the network can be discontinuous, because a sensor may suddenly become illuminated or hidden from the source. Also, the machine cannot go beyond the boundary of the world. If the network output indicates a step out of the world, the machine goes as far as possible and stops. It may wait many iterations before the network generates an acceptable step within the world.

The procedure for studying the effect of sensory input on the trajectories is then quite straightforward. The machine is initially placed at some point, usually the centre of the world with coordinates (0,0), and is initialized with random connections. The weights and offsets may be either random or from the final state resulting from previous iterations. The final state of the network is always stored at the end of a trial so that it can be continued in the same or a different environment. The source is given a constant intensity I, and the simulation is started. Our program gives a graphical animation of the machine, source and world shown in Fig. 3.9. It shows the machine moving in the world.

If the source is *off* ($I = 0$), the machine moves along trajectories such as those shown in section 3.1, depending on the value of s_r. If $s_r < 0.49$, the motion eventually ceases (Fig. 3.2) because the network stabilizes. If $s_r > 0.5$, the motion may appear periodic or chaotic. (Figs. 3.3 - 3.8).

3.2.1 Effect of positive input

An intense source, e.g. with $I = 5.0$, has a *dramatic* effect on the trajectory of the machine driven by a non-chaotic network.

To ensure that the sensor signals dominate input to the nodes in the input region of the recurrent network, we fixed the weights for the connections from the sensors at +10.0 for the trials in this section. Recall that with our updating scheme, the reurrent node weights tend to approach a magnitude of 2.5 (section 3.1.1).

When a machine with a *stable* network, i.e. one that has converged in the absence of a source, is placed at (0,0) and the simulation is started with the source *on*, the machine immediately starts moving, first toward the source, then away, and eventually it is *strongly attracted to the source*. It appears to attach itself to the source for many iterations, but then moves rapidly away, eventually being attracted again.

It is difficult to express the effect of this motion without actually seeing the animation. Figure 3.11 shows 10K iterations of the trajectory of a machine with $s_r = 0.4$ (cf. Fig. 3.2). The source can be seen as a small circle at (-5.6, -5.6). The lines show the trajectory of the machine *centre*, which is 0.5 units from the machine edge. From this it is evident that the machine tends to hover around the source, but the periods when it is actually immobile while touching the source cannot be deduced. However, Fig. 3.12 shows the distance of the machine from the source at each iteration, and from this the attachment periods can be seen. There are also long periods when the machine is immobile at the edge of the world, but the attractions continue to occur.

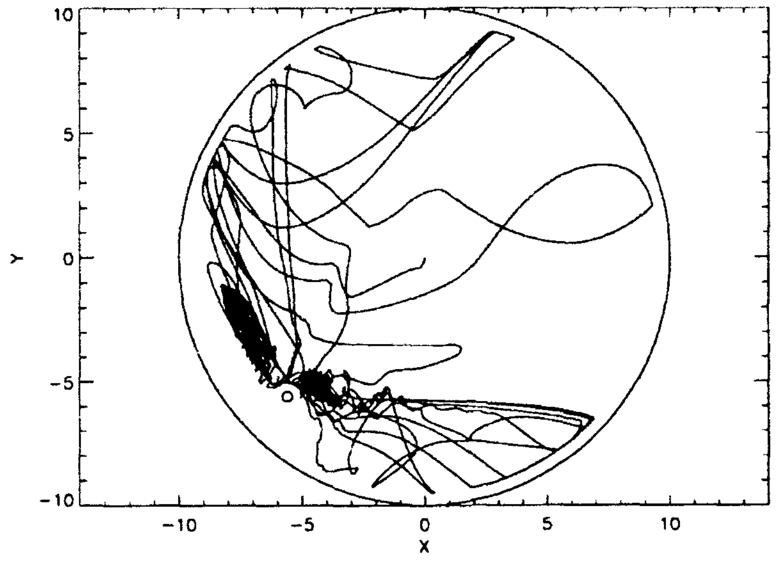

Figure 3.11

Trajectory with $s_r = 0.4$ and source intensity $= 5.0$

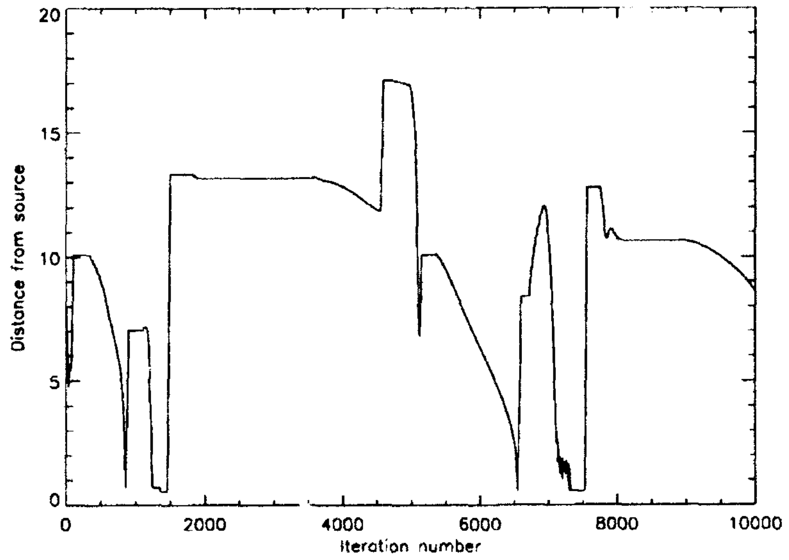

Figure 3.12

Distance from source ($s_r = 0.4$, $I = 5.0$)

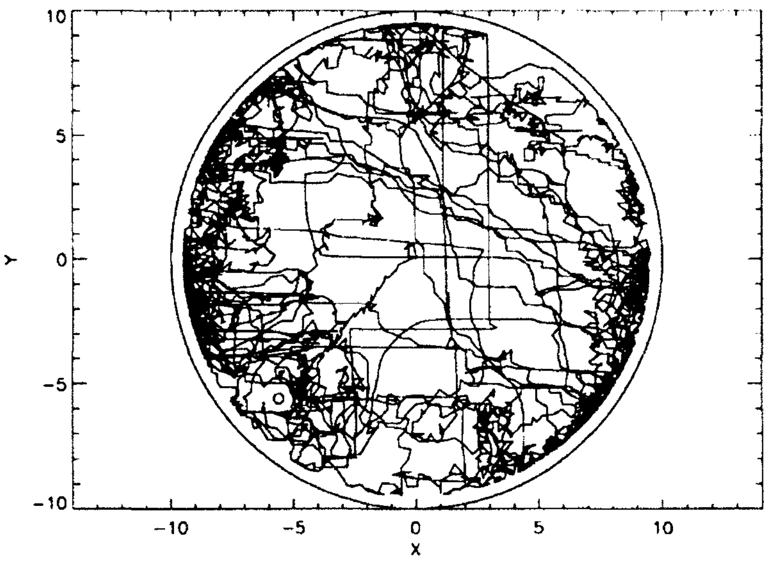

Figure 3.13

Trajectory with $s_r = 1.0$ and source intensity = 5.0

A *chaotic* network is also affected by the presence of the source. With $s_r = 1.0$, the machine is also initially attracted towards the source, but rapidly moves away. It moves much more freely over the entire world (Fig. 3.13), and although there are repeated periods of attraction, the times spent touching the source are brief (Fig. 3.14).

The motion of these systems is very much like that of a moth hovering near a light.

3.2.2 Effect of negative input

A negative input signal may be obtained by reflecting Fig. 3.10 about the x-axis. When this negative signal is fed into the recurrent network, the resulting trajectories are quite different.

For $s_r < 0.5$ the machine now tends to avoid touching the source. Figure 3.15 shows the trajectory with $s_r = 0.4$, and a negative sensor response, again with $I = 5.0$. Comparing this with Fig. 3.11, it can be seen that the machine is much less likely to touch the source. In fact, Fig. 3.15 shows 15K iterations, and only once, after 10K iterations, did the machine actually touch the source.

For values of s_r that are in the chaotic regime, the machine exhibits no particular attraction to the source. Figure 3.16, showing 15K iterations with $s_r = 1.0$, indicates that the machine is more likely to be farther from the source than for the case of positive input (cf. Fig. 3.13)

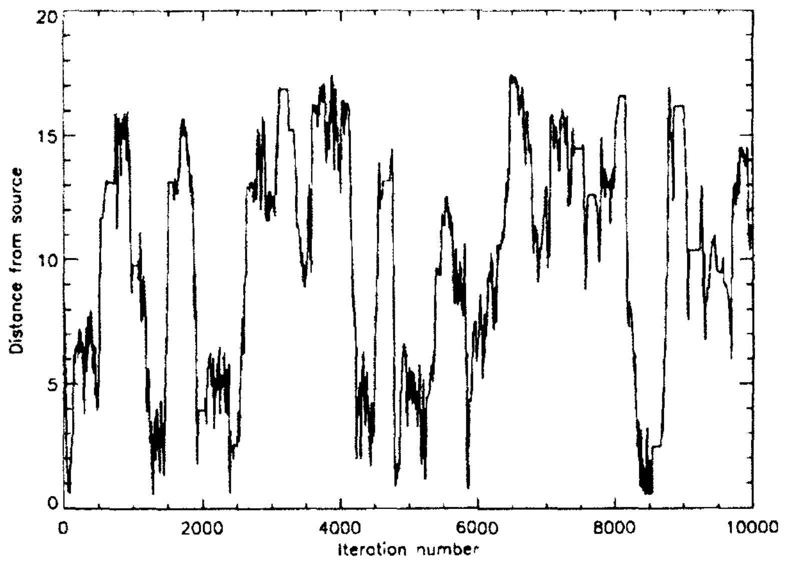

Figure 3.14

Distance from source ($s_r = 1.0$, $I = 5.0$)

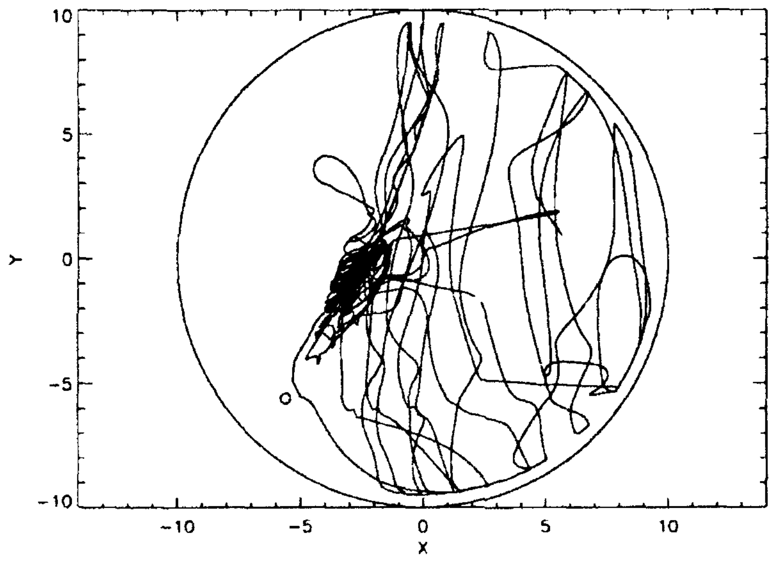

Figure 3.15

Trajectory with negative input ($s_r = 0.4$, $I = 5.0$)

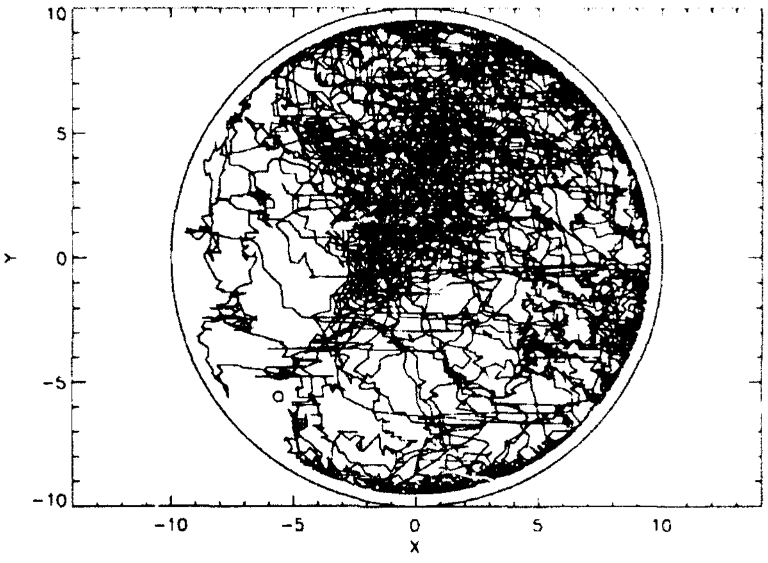

Figure 3.16

Trajectory with negative input (s_r = 1.0, I = 5.0)

A similar effect is obtained if an *inverted* input signal is used; i.e. if the signal is always positive but goes to 1.0 as the intensity approaches zero and goes to zero as the intensity gets large. In this case the machine also appears to avoid the source.

4. Discussion and Conclusions

Iterative signals generated by a recurrent network of sigmoidal nodes with no external input can converge to stable or periodic states, or they may be chaotic. In Section 2.1 we showed that the approach to chaos in a 3-node system may be achieved by the increase in magnitude of a single network parameter, s, which controls the steepness of the node responses. This is consistent with Feigenbaum's model [1] of a nonlinear system driven by a single parameter. As s increases, a stable system becomes periodic, and then exhibits period doubling to chaos. Beyond the first chaotic region, other regions of periodicity and chaos occur, and because of the step-like functions generated by higher s values, a region of erratic periodicity follows.

The behaviour of large recurrent networks with fixed weights and offsets is more difficult to analyse. However, from the results given in Section 2.2, it seems that the node response parameter s can be used to ensure stability. Below a certain value of s, a multi-node network will converge to a stable state. For some choices of weights and offsets, large networks can become chaotic as s increases, and the onset of chaotic signal generation appears to be much more abrupt.

A dynamic recurrent network (DRN), in which the connection weights and node offsets change as a function of the node activities, has the same property. In Section 3.1 we showed that a DRN with no external input can be used to define the trajectories of a simulated machine, and that as s increases these trajectories first spiral to a fixed point, then become quasi-periodic, and are finally completely unstructured (chaotic).

When a DRN is exposed to sensory input, the trajectories of the simulated machine are dramatically altered. The machine shows a strong response to an emitting source (Section 3.2). Depending on the sign of the input signal from the sensors, there is evidence of attractive or repulsive behaviour. The motion of the machine, as seen in a graphical animation, is reminiscent of a moth near a light, with periods of erratic attraction, hovering and repulsion. For nonchaotic values of s, the equations that drive the system could stablilize in two possible positions. One is where the machine is far from the source, so that the input is weak and eventually becomes low and constant, and the machine becomes inactive. This is observed when s is low and the source intensity is weak. Here the sum of the node activities is a minimum. The other possible stable position is where the machine is touching the source, so that the input is also essentially constant, and the sum of the node activities is a maximum.

For strong sources, the equations drive the system to maxima in the sum of the node activities, but these configurations are evidently unstable, because after several network iterations the machine moves away from the source.

The DRN that we have constructed is essentially a *responsive* system. In the absence of a source, it may be made to make what looks like a random walk, using a high value for s. Thus the machine can be made to search its environment. When a source is detected by an input signal from one or more sensors, the value of s may be lowered so that the machine is attracted to the source, if the input is positive. It may also be possible to effect repulsive behaviour by using a negative or inverted input.

So far, our simulated machine has no memory. When a source is illuminated, the system takes some time to respond to it. It may be possible to construct a network that retains a memory of the weight modifications associated with each sensor response, so that when a sensor generates a signal the machine immediately responds. There are many possible variations to the network architecture that must be investigated; e.g. changing the growth and decay rates in the modification rules; increasing the number of nodes and connections per node; and using distinct network regions that have different weight and offset updating schemes.

Our weight modification scheme may be called a *pre-post* modification rule, because it uses the strengths of input signals and the resulting output signal at each node. Another possiblity is a *pre-associative* or *pre-modulatory* scheme in which strong coincident input signals to a node may strengthen their connection weights. Kandel and Hawkins [20] summarize the modification rules that are suspected to occur in biological systems.

It may also be possible to use regions of a network to recognize and respond to stored patterns. Depending on the exact initial state, a multinode net may converge iteratively to one of many possible stable states, and this has been proposed as a mechanism for pattern recognition. It should be possible to avoid iterative chaos in these regions of a network by adjusting the local node steepness parameter. The pattern generated may then drive a motor node response in a predetermined way.

Our initial results on the behaviour of dynamic recurrent networks that are exposed to sensory input show promise for the eventual development of autonomous vehicles that respond adaptively to their environment. By constructing complex recurrent systems with sensory input and motor output, it is possible that intelligent machine behaviour may be created, with no human criteria determining the machine responses. The expectation is that it will be possible to define systems that lead to useful machine behaviour.

References

[1] Feigenbaum, M.J., "Universal Behaviour in Nonlinear Systems", Los Alamos Science, Summer 1980, pp. 4-27.

[2] Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Natl. Acad. Sci. USA, **79** (1982), pp. 2554-2558.

[3] Hopfield, J.J., "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons", Proc. Natl. Acad. Sci. USA, **81** (1984), pp. 3088- 3092.

[4] Bruck, J., "On the Convergence Properties of the Hopfield Model", Proc. IEEE, **78** (1990), pp. 1579-1585.

[5] Skarda, C.A. and Freeman, W.J., "How Brains Make Chaos in Order to Make Sense of the World", Behavioural and Brain Sciences, **10** (1987), pp. 161-195.

[6] Yao, Y. and Freeman, W.J., "Model of Biological Recognition with Spatially Chaotic Dynamics", Neural Networks, **3** (1990), pp. 153-170.

[7] Freeman, W.J., "The Physiology of Perception", Scientific American, February 1991, pp. 78-85.

[8] Chay, T.R., "Complex Oscillations and Chaos in a Simple Neuron Model", Proc. IJCNN, Vol. II, July 1991, pp. 657-662.

[9] Wang, X, "Period Doublings to Chaos in a Simple Neural Network", Proc. IJCNN, Vol. II, July 1991, pp. 333-338.

[10] Kurten, K.E. and Clark, J.W., "Chaos in Neural Systems", Physics Lett., **114A** (1986), pp. 413-418.

[11] Kurten, K.E., "Critical Phenomena in Model Neural Networks", Physics Lett. A, **129** (1988), pp. 157-160.

[12] Riedel, U., Kuhn, R. and van Hemmen, J.L., "Temporal Sequences and Chaos in Neural Nets", Phys. Rev. A, **38**, (1988), pp. 1105-1108.

[13] Sompolinsky, H. and Crisanti, A., "Chaos in Random Neural Networks", Phys. Rev. Lett., **61** (1988), pp. 259-262.

[14] Bauer, M. and Martiennssen, W., "Quasi-Periodicity Route to Chaos in Neural Networks", Europhys. Lett., **10** (1989), pp. 427-431.

[15] van der Maas, H.L.J., Verschure, P.F.M.J. and Molenaar, P.C.M., "A Note on Chaotic Behaviour in Simple Neural Networks", Neural Networks, **3** (1990), pp. 119-122.

[16] Chapeau-Blondeau, F. and Chauvet, G., "Stable, Oscillatory and Chaotic Regimes in the Dynamics of Small Neural Networks With Delay", Neural Networks, **5** (1992), pp. 735-743.

[17] *Neurocomputing, Foundations of Research*, Eds. Anderson J.A. and Rosenfeld, E., MIT Press Cambridge MA. 1988.

[18] *Neurocomputing 2, Directions for Research*, Eds. Anderson J.A. and Rosenfeld, E., MIT Press, Cambridge MA, 1990.

[19] Hebb, D.O., *The Organization of Behaviour*, John Wiley and Sons, Inc., New York, 1949.

[20] Kandel, E.R. and Hawkins, R.D., "The Biological Basis of Learning and Individuality", Scientific American, September, 1992.

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence Research Establishment Suffield Box 4000 Medicine Hat, AB T1A 8K6	2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable) Unclassified

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title.)

The Effect of Sensory Input on Trajectories Generated by Recurrent Neural Networks

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Simon A. Barton

5. DATE OF PUBLICATION (month and year of publication of document) January 1993	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.) 46	6b. NO. OF REFS (total cited in document) 20

6. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Suffield Report 589

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant) 031SC	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)
10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availabilty (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both offical languages unless the text is bilingual).

We show that the iterative signals generated by a simple recurrently connected network of only 3 sigmoidal nodes become stable, periodic or chaotic, depending on the magnitude of a global parameter (*s*) that controls the steepness of the node responses. High values of *s* produce chaotic signals, and low values lead to stable signals, i.e. unchanging from cycle to cycle. The transition from stability, through periodicity to chaos is shown to be consistent with Feigenbaum's model of critical nonlinear systems.

For large networks that are recurrently connected, the stability also depends critically on the node connection weights and biases. When these are fixed randomly, it is possible to find large systems that also show a transition from stability to chaos by the adjustment of *s*. When the weights and biases are allowed to evolve by a modified Hebbian rule, which depends on the internal state of the network, we find that the system becomes stable or chaotic, again depending on *s*, but does not appear to exhibit periodicity.

We have developed a software simulation of a machine whose motion in two dimensions is controlled entirely by the iterative activity of a recurrently connected neural network. The activities of some randomly chosen nodes drive motor responses that control the movement. In the absence of external input, the machine can be made to spiral into inactivity or to follow a chaotic trajectory. If the simulated environment contains one or more emitting sources, and the machine carries sensors whose responses are fed into the network, the motion is influenced by the sensory input. Under a Hebbian-type learning scheme, we show that the machine is strongly attracted to an emitting source when *s* is not in the chaotic regime. The motion of the machine in the region of a source is characterized by cycles of erratic attraction and repulsion, and is highly reminiscent of the motion of a moth near a light.

It is possible that recurrent networks may be used to develop navigation and obstacle avoidance capabilities that are self-learned in an autonomous land vehicle.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Neural Network dynamics

Machine Intelligence

Autonomous land vehicles, guidance and control